Trabajo de Final de Máster

Máster en Ingeniería Industrial
Máster en Ingeniería de la Organización

Gestión de movimientos en suelo en el aeropuerto de Barcelona

MEMORIA

Autor: Pablo Villegas Martín
Director: Lluís Pérez Vidal

Convocatoria: Septiembre, 2018

ETSEIB

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

UPC

Resumen

Uno de los objetivos básicos en el transporte es la seguridad.

Cuando se habla de sistemas de transporte con índices muy elevados de mortalidad en caso de accidente, como son los sistemas aéreos, se deben poner todas las atenciones preventivas necesarias y complementarias que reduzcan al mínimo la probabilidad de que ocurran.

El propósito de este trabajo es incorporar una herramienta más, a todos los medios actualmente usados, que ayude al control y la seguridad en el transporte aéreo. En concreto, se pretende mejorar la seguridad en las maniobras de rodaje, conjunto de movimientos que la aeronave realiza en las carreteras de rodaje del aeropuerto.

En la primera parte del trabajo se realiza una visión global del estado en el que se encuentra el aeropuerto de Barcelona, los sistemas actuales utilizados para el control de los aviones, así como numerosas definiciones útiles para la comprensión del trabajo. Posteriormente se explica el funcionamiento del sistema generado y se detallan los niveles de implementación. A modo de resumen, este sistema consiste en el control a tiempo real de la situación de todos los aviones que se encuentran en el aeropuerto, permitiendo así a los controladores dar las órdenes pertinentes o que de manera automática el avión reciba instrucciones.

Se ha generado una interfaz gráfica capaz de absorber un sistema A-SMGCS de nivel 4.

Finalmente, en el último capítulo, se confecciona un presupuesto que permitirá ver el coste de referencia para la implementación del sistema, así como los beneficios económicos que supone para los *stakeholders* del proyecto.

Índice

1. Glosario

ADS-B (Automatic Dependent Surveillance-Broadcast - Sistema de Vigilancia Dependiente Automática): tecnología de vigilancia cooperativa para el seguimiento de la posición de los aviones vía satélite

AENA (Aeropuertos Españoles y Navegación Aérea)

ATC (Air Traffic Controller)

IATA (International Air Transport Association)

ICAO (International Civil Aviation Organisation): Organización de las Naciones Unidas que estudia los problemas de la aviación civil internacional y promueve reglamentos y normas

CIAIAC (Comisión de Investigación de Accidentes e Incidentes de Aviación Civil)

ULM: Avión ultraligero

TWR: Torre de control

A-CDM (Airport Collaborative Decision Making): es una empresa conjunta entre ACI EUROPE, EUROCONTROL, la Asociación Internacional de Transporte Aéreo (IATA) y la Organización de Servicios Civiles de Navegación Aérea (CANSO), cuyo objetivo es mejorar la eficiencia operativa de todos los operadores aeroportuarios reduciendo los retrasos, aumentando la previsibilidad de los eventos durante el progreso de un vuelo y optimizando la utilización de los recursos

Aeródromo: Área definida de tierra o de agua (que incluye todas sus edificaciones, instalaciones y equipos) destinada total o parcialmente a la llegada, salida y movimiento en superficie de aeronaves

RVR (Alcance visual en la pista): Distancia hasta la cual el piloto de una aeronave que se encuentra sobre el eje de una pista puede ver las señales de superficie de la pista o las luces que la delimitan o que señalan su eje

Área de maniobras: Parte del aeródromo que ha de utilizarse para el despegue, aterrizaje y rodaje de aeronaves, excluyendo las plataformas

Taxiway (Calle de rodaje): Vía definida en un aeródromo terrestre, establecida para el rodaje de aeronaves y destinada a proporcionar enlace entre una y otra parte del

aeródromo

TCL (Línea de eje de calle de rodaje): Línea que permite al piloto de una aeronave el rodaje seguro por el área de movimiento

Pista: Área rectangular definida en un aeródromo terrestre preparada para el aterrizaje y el despegue de las aeronaves

Plataforma: Área definida en un aeródromo terrestre, destinada a dar cabida a las aeronaves, para los fines de embarque o desembarque de pasajeros, correo, carga, reabastecimiento de combustible, estacionamiento o mantenimiento. Incluyendo calles de rodaje, calles de acceso al puesto de estacionamiento y vías de servicio

LVP (Procedimientos de Visibilidad Reducida): Procedimientos específicos que permiten las operaciones en condiciones de visibilidad reducida

Punto de espera de la pista: Punto destinado a controlar el tránsito o proteger una pista en el que las aeronaves en rodaje y los vehículos se detendrán y se mantendrán a la espera, hasta recibir nueva autorización de la Torre de Control del Aeródromo o información de vuelo en caso de aeronaves, o permiso de la dependencia AFIS en caso de vehículos

Vehículo - Guía: Vehículo con luces amarillas anticolisión utilizado para guiar aviones u otros vehículos

Runway: pista donde se realizan las maniobras de aterrizaje y despegue

2. Prefacio

2.1. Motivación

La seguridad en el transporte es un desafío importante y una prioridad para todos.

Si hablamos de la aviación, en los Estados Unidos, se producen un promedio de tres incursiones diarias en la pista de despegue/aterrizaje. Cada uno de estos incidentes puede causar daños significativos, tanto humanos como económicos.

Estas incursiones en la pista de despegue/aterrizaje se producen por unos pocos segundos de distracción y en ocasiones han causado muertes, y es por ello que se convierten en un serio problema de seguridad.

La posibilidad de generar una plataforma que pudiese ayudar a solucionar este tipo de problemas ha despertado un gran interés en mí.

2.2. Requerimientos previos

Los requerimientos para el desarrollo de este trabajo son básicamente: tener conocimientos sobre la aviación, en concreto sobre la normativa y el funcionamiento de los aeropuertos, y conocimientos sobre el lenguaje de programación Python, en concreto pygame.

Los conocimientos sobre Python adquiridos previamente en los estudios de grado han servido de base para el desarrollo de este trabajo y fueron el punto de motivación que me llevó a la elección de la temática.

3. Introducción

3.1. Objetivos del trabajo

El objetivo base del trabajo es generar una plataforma visual que ayude al control de las aeronaves que se encuentran realizando maniobras sobre las pistas de rodaje del aeropuerto.

Este objetivo es solo la cara visible del objetivo real: reducir el número de accidentes en los aeropuertos. Tratando de evitar así tanto los daños humanos como las grandes pérdidas económicas que estos sucesos suponen.

3.2. Alcance del trabajo

El trabajo tiene como objetivo la creación de una plataforma que ayude al control en tierra de las aeronaves que operan en el aeropuerto de Barcelona, en concreto las que tiene asignadas la Terminal 1.

La plataforma está destinada a ser utilizada por los controladores en la torre de control, y no se contempla su incorporación con esta estructura en las aeronaves para uso del piloto.

El alcance del trabajo es la generación de un simulador que opere mediante el intérprete de Python. No se pretende la búsqueda de una plataforma alternativa para su distribución.

Aunque no entra en el alcance del trabajo, se ha tratado de generar un conjunto de algoritmos lo más genéricos posibles, de manera que se facilita una posible futura aplicación a otras terminales del propio aeropuerto o en otros aeródromos.

4. Lenguaje implementación

En este apartado se determina y analiza el lenguaje de programación seleccionado para la ejecución de este trabajo.

Para la selección del lenguaje de programación se han tenido en cuenta las características técnicas y mis conocimientos previos del lenguaje Python, y concretamente de pygame. Esto supone una gran ventaja sobre el resto de los lenguajes de programación ya que se podrá llegar a un nivel de detalle mucho mayor en el programa.

Pygame es un conjunto de módulos del lenguaje Python que permite la creación de interfaces gráficas en dos dimensiones de una manera sencilla. Pygame es libre, publicado bajo la GNU GPL[1]. Con él se pueden crear programas de código abierto.

La programación Pygame está orientada al manejo de sprites. Se trata de un tipo de mapa de bits dibujados en la pantalla del ordenador por un hardware gráfico especializado sin cálculos adicionales de la CPU. A menudo son pequeños y parcialmente transparentes, de este modo, por mucho que el programa los asuma como rectángulos, de manera visual por la interfaz pueden asumir otras formas.

Aunque el lenguaje sea sencillo y permita prototipar y desarrollar rápidamente, los resultados pueden llegar a ser profesionales.

4.1. Análisis lenguaje Python

En este apartado se clasifican las características del lenguaje de programación Python como ventajas o desventajas.

Se debe remarcar que aunque una característica sea clasificada como ventaja o desventaja respecto a otros lenguajes de programación, no significa que Python sea el mejor o el peor en esa característica, sino que simplemente se ha considerado lo suficientemente buena o mala.

[1] La Licencia Pública General de GNU (GNU GPL) es una licencia de derecho de autor ampliamente usada en el mundo del software libre y código abierto, y garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software.

1. Ventajas lenguaje Python:

    1. *Facilidad de uso*: sencillez de escritura/lectura y gestión automática de memoria.
    2. *Legibilidad del código*: estructura bastante natural del código.
    3. *Multiplataforma*: Python es un leguaje de programación interpretado, por lo que funciona en cualquier sistema que integre su interpretador.
    4. *Gran base de usuarios*: existencia de mucho código en internet, que facilita el aprendizaje.
    5. *Lenguaje libre y código abierto*.
    6. *Programación orientada a objetos*: modelar todo en función a clases y a objetos.
    7. *Abundancia de bibliotecas*: extensión de las funcionalidades básicas. En este caso se ha usado Pygame para el desarrollo del programa.

2. Desventajas lenguaje Python:

    1. *Lentitud de ejecución*: el procesador no ejecuta directamente el programa y hace que los tiempos de ejecución sean elevados, a pesar de ser un lenguaje interpretado con una característica multiplataforma considerada como ventaja.

En resumen, sólo se encuentra una desventaja técnica destacable y numerosas ventajas sobre el lenguaje Python. Por tanto, al ser un lenguaje adecuado para este trabajo y teniendo en cuenta los conocimientos previos a la ejecución del mismo, se escoge el lenguaje Python para desarrollar el programa de control de aviones del aeropuerto de Barcelona.

5. Aeropuerto Barcelona

En este capítulo, se explica lo que representa el aeropuerto de Barcelona – El Prat. La información relevante de su estructura, su capacidad para seguir expandiéndose y su espectacular crecimiento durante los últimos años.

El aeropuerto de Barcelona o aeropuerto del Prat se encuentra situado a 12 km al sudoeste de la capital catalana, en el municipio del Prat de Llobregat, a una altura de 3,8 m sobre el nivel del mar.

A lo largo de los años se ha ido posicionando como uno de los principales aeropuertos del sur de Europa, y es por ello que ha tenido que evolucionar hasta su estructura actual que le permite absorber dicho crecimiento.

Es el mayor aeropuerto en extensión y tráfico de Cataluña y el segundo aeropuerto con mayor tráfico de España, detrás del Aeropuerto Adolfo Suárez de Madrid-Barajas. A continuación, se detallan las infraestructuras actuales y cómo se ha llegado hasta ellas, así como posibles reformas en el futuro.

5.1. Infraestructuras

5.1.1. Evolución desde 1916 [2]

- 1916: Primeras instalaciones situadas en la granja La Volatería

Desde la creación del aeropuerto de El Prat en 1916, el crecimiento en cuanto a número de pasajeros fue lento.

- 1948: Construcción de una segunda pista, la 07-25
- 1990: Se efectuaron diversos cambios en previsión a los Juegos Olímpicos de Barcelona 1992

En el año 1992, con la llegada de las olimpiadas, el tráfico experimentó un fuerte crecimiento, superándose los 10 millones de pasajeros. La mejora de técnicas en la zona de operaciones del aeropuerto en 1994, con la incorporación del ILS[2] para realizar

[2] El sistema de aterrizaje instrumental (o ILS, del inglés: Instrument Landing System) es el sistema de ayuda a la aproximación y el aterrizaje establecido por OACI (Organización de Aviación Civil

aterrizajes con instrumentación, ayudó a ampliar la capacidad de 38 operaciones por hora a 50. También la liberalización del transporte aéreo en 1995 contribuyó a que ese crecimiento fuera más marcado.

- 2003: Reforma de la terminal B y ampliación de la terminal A

La presencia de nuevas aerolíneas exigió que se fueran abriendo nuevas rutas. En 2003 se realizó un ambicioso Plan Director para realizar la construcción de la tercera pista del aeropuerto. Con ello se aumentaron las operaciones por hora, ayudado en parte por la ampliación de la pista 07L-25R tanto en longitud como en anchura. Pero posteriormente surgió un problema ya que las pistas podían absorber mayor número de vuelos, pero los aviones no podían acceder a la terminal por congestión tanto en rampa como en la terminal T2. En el mismo año 2003, se vio la necesidad futura de construir una nueva terminal, originalmente llamada Nueva Terminal Sur. No sólo se había ideado ampliar la superficie, sino también en el campo de vuelos, fingers, cintas de equipaje y otros elementos para poder atender a un mayor número de pasajeros.

- 16 de junio de 2009: Inauguración de una nueva terminal de pasajeros, la Terminal 1

La inauguración de la nueva terminal se realizó en 2009, 6 años después de que se presentara el plan director. El aeropuerto actualmente tiene una capacidad de operar hasta 90 vuelos por hora y gestionar un total de 55 millones de pasajeros al año.

5.1.2. Actuales [1]

El aeropuerto de Barcelona dispone de dos terminales: T1 y T2. Las dos terminales suman un total de 268 mostradores de facturación y 64 pasarelas de embarque.

El aeropuerto tiene actualmente tres pistas de despegue y aterrizaje. Dos de ellas en paralelo, denominadas 07L / 25R y 07R / 25L, y una cruzada llamada 02/20 (prácticamente en desuso debido al cruce con las otras dos pistas). El nombre de las pistas está compuesto por un número que marca la dirección en grados (redondeado a la decena más cercana y recortado en el último dígito) con respecto al norte magnético a la que se encuentra dirigida la pista, por tanto, el otro extremo de la pista tendrá una diferencia de 180° y en consecuencia diferirá en 18 puntos en el nombre. La letra que acompaña el 07/25 sólo se da en pistas que transcurren paralelamente, y que por ello están identificadas con el mismo número. Se añade una R (del inglés Right) en la pista derecha, y una L (de Left) en la pista izquierda.

Internacional), permite que un avión sea guiado con precisión durante la aproximación a la pista.

El conjunto de reformas realizadas hasta la fecha de hoy han tenido como objetivo adecuar el aeropuerto al crecimiento de tráfico ocurrido en los últimos años, pasando de los 30 millones de pasajeros en 2008 a ser capaz de gestionar hasta 55 millones.

5.1.3. Situación futura

En 2010 se ordenó al despacho de arquitectura de Ricardo Bofill un proyecto para la construcción de una terminal satélite. Pero cuando en los años consecutivos se vio que la T1 era capaz de absorber la demanda de pasajeros y el ritmo de crecimiento se ralentizaba, se decidió no llevar a cabo la construcción de la terminal satélite hasta que el tráfico no aumentase. Hoy en día el proyecto se encuentra suspendido, a la espera de que se requiera dicha capacidad adicional, aun así, en los próximos años se espera que se inicie el ambicioso proyecto de la terminal satélite si no es posible distribuir el tráfico entre las instalaciones ya existentes.

La nueva terminal estará localizada justo a los pies de la actual torre de control. Su diseño en forma de pájaro se ha estudiado para que no afecte al campo de visión de la torre de control. La terminal se ha ideado para que sea una extensión de la actual T1 y su acceso se realizaría mediante un tren ligero que conectaría las terminales T1 y la terminal satélite. La capacidad que se espera abordar con esta nueva terminal será de 25 millones de pasajeros al año, llegando a aumentar la capacidad total del aeropuerto hasta los 80 millones de pasajeros al año. Dicha capacidad quedaría desglosada de la siguiente manera: la T2 con 27 millones de pasajeros al año, la T1 con 25 millones de pasajeros/año y la T1 satélite con 25 millones de pasajeros/año.

Con los datos de los últimos años (ver figura 5.1), 47,3 millones de pasajeros durante el 2017 y una proyección de aumento de 3 millones de pasajeros al año, se podría alcanzar el límite actual de 55 millones en unos 3 años, que haría necesaria la ampliación de la T1 con la terminal satélite.

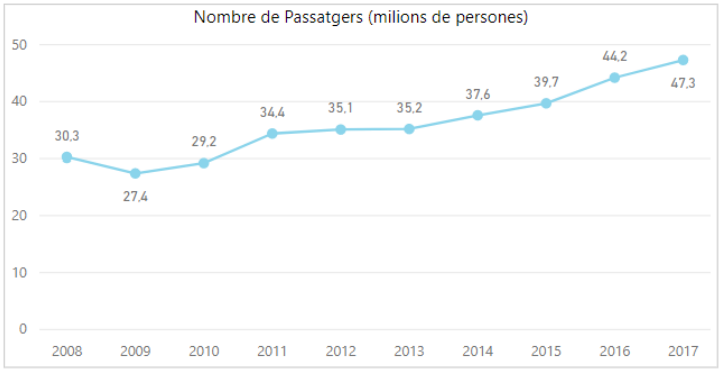


Figura 5.1 Evolución número de pasajeros aeropuerto Barcelona

6. Carreteras de rodaje (*Taxiways)*

Una calle de rodaje o pista de rodaje (del inglés *taxiway*) es la parte de las infraestructuras de un aeropuerto que permite conectar las zonas de hangares y terminales con las pistas de despegue/aterrizaje.

En los aeropuertos de mucho tráfico, como es el caso del aeropuerto de Barcelona, se construye una pista paralela a la pista de aterrizaje con el fin de evitar que tanto los aviones que despegan, como aquellos que aterrizan, ocupen la pista innecesariamente (en tránsito hacia la terminal después de aterrizar o hacia la cabecera de la pista antes del despegue). De esta manera los aviones después de aterrizar pueden abandonar la pista de aterrizaje apenas disminuyan su velocidad y circular hacia la terminal por fuera de la misma, permitiendo que otros aviones puedan utilizar la pista de aterrizaje antes. Esto permite aumentar la capacidad (número total de operaciones) que pueden lograrse en la pista de aterrizaje.

Para mejorar aún más la capacidad de las pistas de aterrizaje, se suelen construir salidas rápidas de la misma que permiten que los aviones la abandonen a velocidades más altas. Esto facilita que el avión desocupe la pista más rápidamente, posibilitando que otros aterricen en un espacio más corto de tiempo. En la siguiente se caracterizan los elementos descritos anteriormente en el aeropuerto que está siendo objeto de estudio, el aeropuerto de Barcelona:

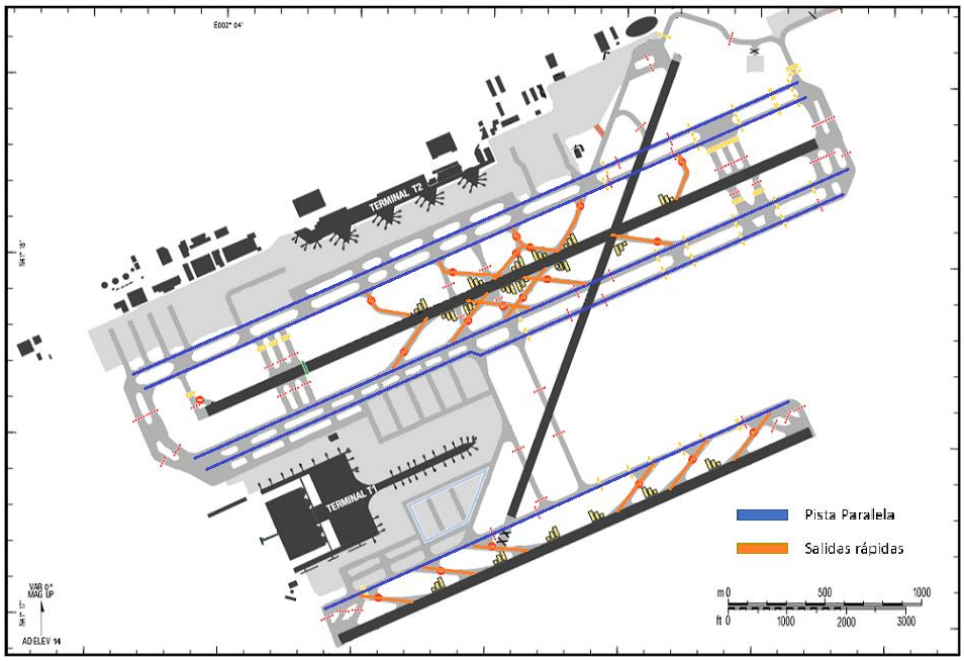


Figura 6.1 Diagrama Aeropuerto Barcelona. Salidas rápidas y pistas paralelas.

Por tanto, todas las vías grises observadas en la figura 6.1 serán las que permitan el enlace entre las pistas y las terminales o hangares.

A la hora de realizar la operación de taxi se deberán tener en cuenta también los “puntos calientes” del aeropuerto. Un "punto caliente" suele ser una zona donde se produce un cruce complejo o confuso de calle de rodaje / calle de rodaje o calle de rodaje / intersección de pista. Esta área de mayor riesgo tiene un historial o potencial para incursiones en la pista o incidentes. Las causas que pueden provocar dichos incidentes son: la disposición del aeropuerto, el flujo del tráfico, el marcado del aeropuerto (señalización e iluminación).

Se debe prestar especial atención a cualquier intersección compleja o áreas designadas en el diagrama del aeropuerto como "puntos calientes" para reducir el riesgo de una incursión en la pista. El diagrama del aeropuerto de Barcelona contempla 6 “puntos calientes” como se ve en la siguiente figura 6.2

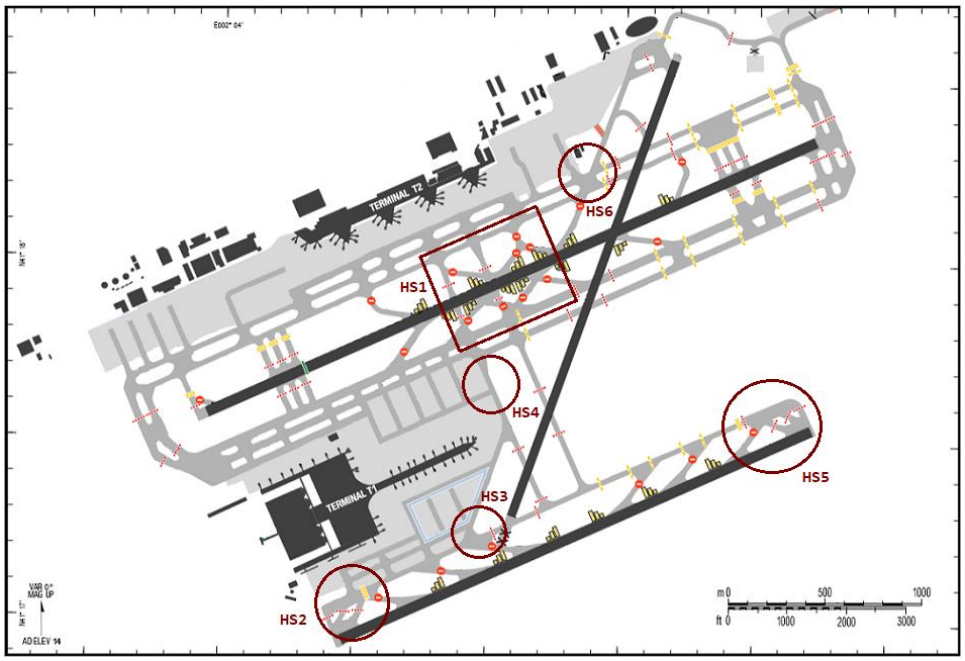


Figura 6.2 Diagrama Aeropuerto Barcelona. Puntos calientes.

Debido al gran número de elementos que se están moviendo simultáneamente en el aeropuerto de Barcelona por los *taxiways,* las limitaciones de número de vías de rodajes y los “puntos calientes” se deben adecuar a las rutas de los diferentes aviones para evitar poner en riesgo la seguridad de los agentes implicados.

Uno de los problemas que se pueden dar son las incursiones en la pista, definidas formalmente por la FAA como "cualquier ocurrencia en un aeródromo que involucre la presencia incorrecta de un avión, vehículo o persona en el área protegida de una superficie designada para el aterrizaje y despegue de la aeronave". Estas incursiones suelen tener dos tipos de causas principalmente:

1. Desviaciones del piloto:

   • Cruzar una marca de espera de la pista sin autorización de ATC

   • Despegar sin autorización

   • Aterrizar sin autorización

2. Incidentes operacionales (OI):

   • Despegar una aeronave en una pista mientras otro avión está aterrizando en la misma pista

• Emisión de una autorización de despegue mientras la pista está ocupada por otro avión o vehículo

Según datos de la FAA, aproximadamente el 65 por ciento de todas las incursiones en la pista son causadas por los pilotos.

Es por todo ello que se toma especial atención en las operaciones de rodaje y se sigue un procedimiento con control exhaustivo.

6.1. Procedimiento rodaje

Una vez se han visto los peligros en los procedimientos de taxi, es importante determinar las normas que deben seguir los pilotos en las operaciones de taxi. En este apartado se detalla el procedimiento habitual en una operación de taxi, así como el lenguaje específico usado por la ATC para dar las instrucciones durante el rodaje, y los procedimientos que se deben seguir según la instrucción.

6.1.1. Conciencia Situacional

Conciencia Situacional (SA) significa entender lo que está pasando a tu alrededor. No solo se trata de la recopilación de información, sino que requiere reunir la correcta información, analizarla y tomar decisiones adecuadas según lo analizado.

En los aeropuertos con plataforma, la superficie del aeropuerto se divide en dos partes:

- *El área sin movimiento*: se define como rampas y delantales que no son controlados por ATC, lo que significa que las aeronaves pueden moverse o tomar una vía de taxi sin autorización ni comunicaciones con la Torre de control.

- *El área de movimiento*: se define como todas las calles de rodaje y pistas. Están bajo la jurisdicción de la torre de control, por lo que todo avión requiere autorización antes de ingresar en dicha zona. El límite entre la rampa y las calles de rodaje se define por dos líneas amarillas: una sólida y otra discontinua.

A modo de resumen, en las áreas de no movimiento el piloto se rige por su criterio según su inspección visual de la zona; y en las áreas de movimiento se rige por las instrucciones dadas desde la torre de control (siempre evaluando su viabilidad de manera visual). En caso de tener dudas sobre la seguridad de una maniobra ordenada desde ATC, se debe detener el avión y volver a contactar con la torre de control para explicar la nueva situación y de este modo verificar o modificar la instrucción previa.

Cuando no se está familiarizado con el aeropuerto o no se está seguro de la ruta de un taxi, se puede solicitar a ATC un "Taxi progresivo". El taxi progresivo requiere que el controlador proporcione instrucciones de taxi paso a paso. La decisión final de actuar según las instrucciones de ATC recae sobre el piloto por inspección visual. Si no se puede cumplir con seguridad con cualquiera de las instrucciones de ATC se debe informar de inmediato usando la palabra "UNABLE".

6.1.2. Lenguaje ATC

6.1.2.1. Lenguaje ATC: “Hold Short”

Es el signo y marca más importante en el aeropuerto. Este se encuentra en un trozo de calle de rodaje que conduce directamente a una pista de aterrizaje/despegue. Ellos representan la posición o la ubicación donde la aeronave debe detenerse para no entrar en el entorno de la pista.

En un aeropuerto con torre de control, las marcas de espera en la pista nunca se cruzarán sin instrucciones de la ATC explícitas. No se podrá ingresar a una pista en un aeropuerto con torre de control a menos que las instrucciones dadas por ATC sean: cruzar, despegar, o "alinearse y esperar" en esa pista específica.

Las instrucciones para cruzar una pista normalmente se emiten una sola vez, y una aeronave debe haber cruzado la pista anterior, antes de que se emita otro cruce de pista. Esto se aplica a todas las pistas: tanto las activas y las inactivas como las cerradas. En el aeropuerto de Barcelona se deberá dar la instrucción durante el día para la pista 02/20, por mucho que en ese horario esté inactiva.

6.1.2.2. Lenguaje ATC: “Line up and wait”

La instrucción ATC para "alinearse y esperar" no es una autorización para el despegue. Solo es una autorización para ingresar en la pista y mantenerse en posición para el despegue en espera de la autorización.

Algunas razones por las que se da esta instrucción y se retrasa la autorización de despegue pueden ser:

- Otro avión se encuentra aterrizando o despegando
- Se hayan despertado turbulencias o situaciones meteorológicas adversas
- Tráfico cruzando la pista

6.1.2.3. Lenguaje ATC: “Runway shortened”

Si la pista de aterrizaje o despegues es más corta de lo que indica el diagrama del aeropuerto por condiciones excepcionales (obras u otras causas) se debe comunicar al piloto.

6.1.2.4. Lenguaje ATC: “Cleared for takeoff”

Instrucción que indica que la pista sobre la que se realizará el despegue está libre y por tanto se concede permiso al avión para realizar la maniobra.

6.1.2.5. Lenguaje ATC: “Cleared to land”

Instrucción que indica que la pista sobre la que se realizará el aterrizaje está libre y por tanto se concede permiso al avión para realizar la maniobra.

7. Sistemas de control aviación

El conjunto de sistemas utilizados para el control de la aviación contiene sistemas de navegación y comunicación por satélite (comunicación, navegación, vigilancia y gestión del tráfico aéreo). Estos sistemas son conocidos como CNS/ATM (Communication Navigation Surveillance / Air Traffic Management).

En concreto, se estudiarán los sistemas que se utilizan para el control en tierra pues es el tipo de control que entra en el alcance de este proyecto.

El control de tierra es responsable de las áreas de "movimiento" del aeropuerto. Incluye todas las calles de rodaje, pistas inactivas, áreas de espera, y algunas intersecciones de transición donde llegan las aeronaves después de haber desocupado la pista o la puerta de salida.

Cualquier aeronave, vehículo o persona que camine o trabaje en estas áreas debe tener autorización del control de tierra. Esto se hace normalmente a través de la radio VHF/UHF, pero puede haber casos especiales donde se usan otros procedimientos. Las aeronaves o vehículos sin radios deben responder a las instrucciones ATC a través de señales luminosas de aviación.

Algunos aeropuertos con mayor actividad tienen un radar de movimiento de superficie (SMR), diseñado para mostrar aviones y vehículos en tierra. Estos son utilizados por el control de tierra como una herramienta adicional para controlar el tráfico en tierra, particularmente de noche o con poca visibilidad. Hay una amplia gama de capacidades en estos sistemas.

Figura 7.1 Visualización del sistema de radar terrestre, aeropuerto de Schiphol (Amsterdam).

Actualmente la Unión Europea está aportando apoyo financiero en distintas iniciativas orientadas a mejorar la gestión del tráfico en el espacio aéreo europeo. El proyecto *Intervuse* se puso en marcha con el objetivo de desarrollar un nuevo sistema de guía y control del movimiento en la superficie (SMGCS) que resultase rentable y mejorase los puntos débiles de las tecnologías de SMGCS existentes. El nuevo sistema del que se habla es el A-SMGCS.

7.1. Sistemas actuales aeropuerto Barcelona [7]

En la siguiente tabla podemos ver los sistemas de que dispone la torre de control (TWR) para realizar sus tareas de control:

Tabla 7.1 Sistemas Control Aeropuerto Barcelona

Systems		Available in TWR
Surface Movement Radar (SMR)	Basic SMR	Yes
	A-SMGCS (level 1)	Yes
	A-SMGCS (level 2)	No
Use of downloadable Mode-S link		No
Electronic Strips		No
AMAN		No
DMAN		Yes
Spacing support tools in OPS		No

7.1.1. A-SMGCS

Sistema Avanzado de Guía y Control de Movimientos en Superficie se compone de una combinación de sistemas que proporcionan servicios a aeronaves y vehículos con el fin de mantener el máximo rendimiento del aeropuerto bajo todas las condiciones climáticas posibles, al tiempo que se mantiene el nivel de seguridad requerido.

Las ventajas que proporciona este tipo de sistema:

- Proporcionar una representación del tráfico real del aeródromo en una pantalla, independientemente de la conexión de línea de visión entre el controlador y el móvil (aviones, vehículos auxiliares, etc...).

- Proporcionar la posición y la identidad de todos los móviles cooperativos, dentro del volumen de cobertura independientemente de las condiciones de visibilidad y la línea de visión del Controlador.

- Ayudar al controlador a evitar colisiones entre todas las aeronaves y vehículos, especialmente en condiciones en las que no se puede mantener el contacto visual.

- Detectar e indicar la posición de posibles intrusos.

Algunos de los servicios que puede prestar dicho sistema son:

- Servicio de soporte de seguridad aeroportuaria. Las funciones que proporciona esto son:

    - Monitoreo de pista y alerta de conflicto (RMCA): es una herramienta de alerta de conflictos para controladores que monitorea los movimientos en superficie del aeropuerto y detecta conflictos entre aeronaves y otros móviles. Utiliza datos de vigilancia y los estudia siguiendo unas reglas y parámetros predefinidos.

    - Conflicto de autorizaciones de ATC (CATC): proporciona una alerta cuando el controlador ingresa una autorización electrónica a través de la HMI, que de acuerdo con un conjunto de reglas acordadas localmente no está permitida desde el punto de vista operacional y de seguridad, es decir, se intenta dar una autorización que no es compatible con alguna de las autorizaciones electrónicas previamente ingresadas.

    - Monitoreo de conformidad de alertas para controladores (CMAC): proporciona a los controladores alertas apropiadas cuando el A-SMGCS detecta la no conformidad con los procedimientos o las autorizaciones para el tráfico en las pistas, calles de rodaje y en el área de la plataforma.

- El servicio de enrutamiento que genera trayectorias de tierra para móviles. Es un habilitador clave para el Servicio de Orientación y algunos elementos del Servicio de Apoyo de Seguridad Aeroportuaria (especialmente la Alerta de Desviación de Ruta).

Genera rutas individuales para móviles basadas tanto en parámetros y restricciones propias del aeródromo como en instrucciones del controlador. Posteriormente se calculan los tiempos de taxi precisos basados en la ruta y estos tiempos pueden ser utilizados por la plataforma A-CDM[3].

- El servicio de orientación que proporciona las funciones:
    - Conmutación automática de las luces centrales de las pistas (TCL)
    - Conmutación automática de barras de parada y activación automática de sistemas avanzados de orientación de acoplamiento visual (A-VDGS)

El sistema ofrecerá diferentes funciones de las anteriormente citadas según el nivel de implementación. Se pueden diferenciar cuatro claros niveles:

- Nivel 1: Procedimientos y vigilancia mejorados, que cubren el área de maniobras para vehículos terrestres y el área de movimiento de aeronaves. Los procedimientos se refieren a la identificación y emisión de instrucciones ATC y autorizaciones. A los controladores se les da información de posición de tráfico e identidad, lo cual es un importante paso adelante de la imagen tradicional de radar de movimiento superficial (SMR).
- Nivel 2: agrega redes de seguridad que protegen las pistas y áreas designadas y los procedimientos asociados. Se generan las alertas apropiadas para los controladores en caso de conflictos entre vehículos en las pistas y la incursión de la aeronave en áreas restringidas designadas.
- Nivel 3: implica la detección de todos los conflictos en el área de movimiento, así como una mejor orientación y planificación para el uso por parte de los controladores.
- Nivel 4: proporciona resoluciones para todos los conflictos. Realiza la planificación y orientación de manera automática para pilotos y controladores.

7.1.2. DMAN

DMAN es un sistema de planificación para mejorar los flujos de salida en los aeropuertos mediante el cálculo del tiempo de despegue objetivo (TTOT) y el tiempo de aprobación de puesta en marcha objetivo (TSAT) para cada vuelo, teniendo en cuenta múltiples

[3] Airport Collaborative Decision Making (A-CDM). Sistema de toma de decisiones.

restricciones y preferencias.

El DMAN tiene varios beneficios que pueden variar dependiendo de las condiciones locales del aeropuerto. En general, los principales beneficios son:

- Medio ambientales: Las aeronaves pueden reducir su tiempo total de rodaje desde su puerta / puesto hasta su despegue real desde la pista y minimizar sus emisiones de combustible.

- Previsibilidad: DMAN agrega previsibilidad al flujo de salida, lo que resulta en una gestión de recursos optimizada para servicios de escala, operadores de líneas aéreas, etc. Al crear una secuencia de salida estable, los diferentes operadores pueden planificar su trabajo de manera más eficiente.

- Reactividad: DMAN permite una reactividad mejorada de eventos imprevistos y, por lo tanto, puede reducir el impacto negativo de tal evento (por ejemplo, cierre instantáneo de la pista debido a restos sospechosos, cambios climáticos, etc.). Al monitorear y distribuir constantemente la información, el DMAN puede apoyar al ATC en un aeropuerto para hacer un nuevo plan basado en la situación actual y distribuir rápidamente la información a los diferentes operadores e interesados en el aeropuerto.

- Seguridad: Durante las operaciones de la hora pico, el DMAN ha tenido un impacto positivo en la carga de trabajo del ATC ya que el DMAN mantiene el avión en la puerta de embarque y, como resultado, reduce la cantidad de tráfico rodado simultáneamente.

7.2. Sistemas futuros aeropuerto Barcelona

Aunque el aeropuerto de Barcelona no da a conocer sus planes de futuro, se cree que el rumbo debe ser hacia la implementación del modo S y de niveles superiores de A-SMGCS.

La implantación del modo S permite que el radar interrogue aeronave a aeronave de modo que el radar puede manejar un escenario de tráfico mucho mayor, evitando así que cada interrogación realizada reciba las respuestas de todos los transpondedores dentro del alcance de la antena y del ángulo de emisión.

Las respuestas de los transpondedores en modo C tienen una resolución de cien pies de altura, mientras que el dato de altura en la respuesta de Modo S tiene una resolución de 25 pies. Así pues, la implantación del Modo S permite el uso de aerovías más próximas en

altitud y por tanto la optimización del uso del espacio aéreo

Como resultado de esta previsión, la ejecución de este proyecto consistirá en la realización de una interfaz gráfica que permita la implementación del nivel 4 de A-SMGCS, es decir, permita la implantación y desarrollo de todas las funciones descritas en el apartado 8.

8. Actores implicados

En este punto se describe el rol y las funciones de cada uno de los actores implicados o relacionados con el sistema A-SMGCS. Para poder entender con claridad el papel que desempeña cada uno de ellos se describen las funciones y tareas de estos. Seguidamente se explica el papel del actor con el sistema actual (A-SMGCS nivel 1) y para finalizar se exponen las diferencias que se producirían con el aumento de nivel del sistema A-SMGCS a nivel 2.

ATC Controladores

Los controladores son el agente principal de esta herramienta.

La función del ATC es administrar las aeronaves y movimientos de los vehículos que se encuentran en el área de maniobras. Estos movimientos deben ceñirse a la planificación, respetando la seguridad en cada uno de los movimientos. Hay cinco tipos de controladores pero solo entran dentro del estudio de este trabajo dos de ellos:

- El controlador de Tierra (GND): encargado de guiar a la aeronave "en tierra" por las calles de rodaje, tanto desde las puertas de embarque a la pista de aterrizaje activa como a otras plataformas en el aeropuerto, o viceversa.

- El controlador de Torre (TWR): tiene al mando la pista o pistas de aterrizaje y las intersecciones; autoriza a la aeronave para aterrizar o despegar. Proporciona información sobre meteorología adversa, trabajos que afecten a la pista y otros elementos que puedan ser un inconveniente para las aeronaves (tales como bandadas de aves).

Sus funciones principales se han explicado con anterioridad, a continuación, se realiza una lista de funciones o acciones más genéricas que deben realizar:

- Identificación de móviles (aeronaves, vehículos auxiliares…)
- Emisión de autorizaciones e instrucciones a todos los móviles participantes
- Monitoreo de la ejecución de las autorizaciones
- Monitoreo de la situación del tráfico en el área de maniobras
- Información de pilotos / conductores sobre el tráfico que rodea su aeronave / vehículo
- Orientación de los móviles participantes
- Alerta a los móviles participantes

El papel del controlador no cambia realmente con la implementación de A-SMGCS nivel 1.

Sí se debe destacar que las tareas citadas anteriormente evolucionan en el sentido de que el servicio de vigilancia proporciona al controlador una nueva fuente de datos sobre la situación del tráfico en todas las condiciones de visibilidad. Esta nueva fuente de datos complementa e incluso podría reemplazar las fuentes habituales de datos de tráfico.

Por lo tanto, con el nivel 1 del sistema A-SMGCS instalado, la posición e identidad de los móviles es proporcionada por A-SMGCS al controlador para ayudarlo a realizar su tarea de control. El controlador utiliza esta información de vigilancia de la siguiente manera:

- El controlador analiza la vista global de la situación del tráfico
- El controlador se enfoca en áreas particulares del aeropuerto o en vehículos que requieren su atención
- La posición de todos los móviles permite al controlador detectar intrusos, o vehículos participantes sin autorización
- La identificación del móvil a través de su etiqueta permite que el controlador se comunique de manera más efectiva
- Las posiciones de los dispositivos móviles con respecto al diseño del aeropuerto ayudan al controlador a establecer una planificación de tráfico y brindar mejor orientación a los pilotos y conductores
- El controlador monitorea en la pantalla que los móviles siguen las autorizaciones que se le han indicado desde la torre de control
- La posición móvil en comparación con el estado de las áreas del aeropuerto permite al controlador anticipar incursiones en áreas restringidas y alertar al móvil
- La posición móvil en comparación con la posición de otros móviles ayuda al controlador a evitar colisiones con otros móviles, alertándolos con suficiente antelación

En cuanto al Nivel 1, el rol del controlador realmente no cambia por la implementación de A-SMGCS Nivel 2, pero las tareas del controlador evolucionan. A-SMGCS de nivel 2 proporciona al controlador una fuente de información de alerta en todas las condiciones de visibilidad, y por tanto será función del controlador aplicar los procedimientos adecuados a cada tipo de alerta.

Esta nueva fuente de datos complementa la detección de conflictos / infracciones realizado por el controlador en el análisis visual del tráfico o utilizando datos de vigilancia.

El controlador usa las alertas para:

- Resolver conflictos / infracciones
- Anticipar incursiones en pistas y áreas restringidas, alertando a los respectivos móviles

- Anticipar el riesgo de colisión entre los móviles en las pistas o calles de rodaje

Cuando es alertado por una detección de conflicto o infracción, esta herramienta de vigilancia permite al controlador comprender la situación puesto que ve en pantalla el área donde se ha dado la alerta y los vehículos implicados. De esta manera, el controlador puede tomar rápidamente las medidas apropiadas para resolver el conflicto o infracción.

Piloto

En la situación actual de SMGCS, el rol del piloto es navegar su avión siguiendo las instrucciones y autorizaciones de ATC provistas a través de R / T, y con la ayuda de ayudas visuales y ATC. Las principales tareas relacionadas con SMGCS son las siguientes:

- Reportar su posición al ATC por R / T;
- Monitorear el tráfico circundante para prevenir la colisión por medios visuales e información de tráfico proporcionada por ATC.

El rol del piloto no cambiará con la implementación del nivel A-SMGCS 1. Al contrario que el controlador, las tareas anteriores no evolucionarán ya que el piloto no tendrá acceso al servicio de vigilancia.

Sin embargo, el uso de un A-SMGCS nivel 1 tendrá el siguiente impacto en el trabajo piloto:

- **Reducción de los reportes por R/T**

  Dado que el controlador conoce la posición y la identidad de la aeronave proporcionada por A-SMGCS, es posible que algunos informes de posición de la aeronave no sean necesarios nunca más.

- **Verificación sensores de vigilancia**

  Dado que se supone que los aviones deben proporcionar su identidad a través de cooperativas sensores de vigilancia, tripulación debe verificar que este equipo funciona satisfactoriamente a bordo y debe usarlo de la manera correcta.

Conductor de vehículos

En la situación actual de SMGCS, la función del conductor es conducir su vehículo siguiendo las instrucciones y autorizaciones ATC proporcionadas por R / T, mediante las referencias visuales. Las principales tareas relacionadas con SMGCS son las siguientes:

- Reportar su posición al ATC por R / T;

- Monitorear el tráfico circundante para prevenir la colisión por medios visuales y la información de tráfico proporcionada por ATC.

En cuanto al piloto, el rol del conductor no cambiará con la implementación de A-SMGCS nivel 1. Sin embargo, cuando el controlador conoce la posición e identidad del vehículo proporcionado por A-SMGCS, es posible que algún vehículo los informes de posición ya no son necesarios. Esto debe confirmarse en procedimientos relacionados con el uso de A-SMGCS.

Además, si el vehículo está equipado para A-SMGCS, por ejemplo con un transpondedor, el conductor debe verificar que el equipo esté activado y funcione de manera correcta.

En A-SMGCS Nivel 2, el conductor del vehículo puede proporcionarse opcionalmente con un servicio de orientación. El papel del conductor del vehículo no cambiará realmente cuando esté equipado con el servicio de orientación. Sus tareas evolucionarán en el sentido de que el servicio de orientación proporcionará al conductor una nueva fuente de datos sobre su posición relacionada con el diseño del aeropuerto en todas las condiciones de visibilidad.

El conductor usará esta nueva información (posición de su vehículo, mapa del aeropuerto, puntos de referencia en el mapa, visualizados en una pantalla) para la navegación de su vehículo. Por ejemplo, cuando esté perdido, no hay indicaciones sobre su posición afuera, o él no puede verlos debido a la visibilidad reducida, él podrá usar la información provista por la guía de servicio para saber exactamente dónde está en la plataforma del aeropuerto.

Otros

Si no es automático, se necesitan uno o más operadores para actualizar las condiciones externas requeridas por A-SMGCS, esto incluye:

- Datos MET, condiciones de visibilidad (incluida la transición entre visibilidad 1, 2, 3 y 4);
- Configuración del aeropuerto: pista en uso, calles de rodaje abiertas…
- Lista de móviles participantes, etc.

Un requisito previo para una detección eficiente de conflictos / infracciones (nivel 2 de A-SMGCS) es la configuración correcta de la herramienta automatizada, es decir, a través de la provisión de la siguiente información:

- Configuración del aeropuerto: pistas en uso, estado de las pistas, áreas restringidas, etc.
- Procedimientos y métodos de trabajo aplicados.

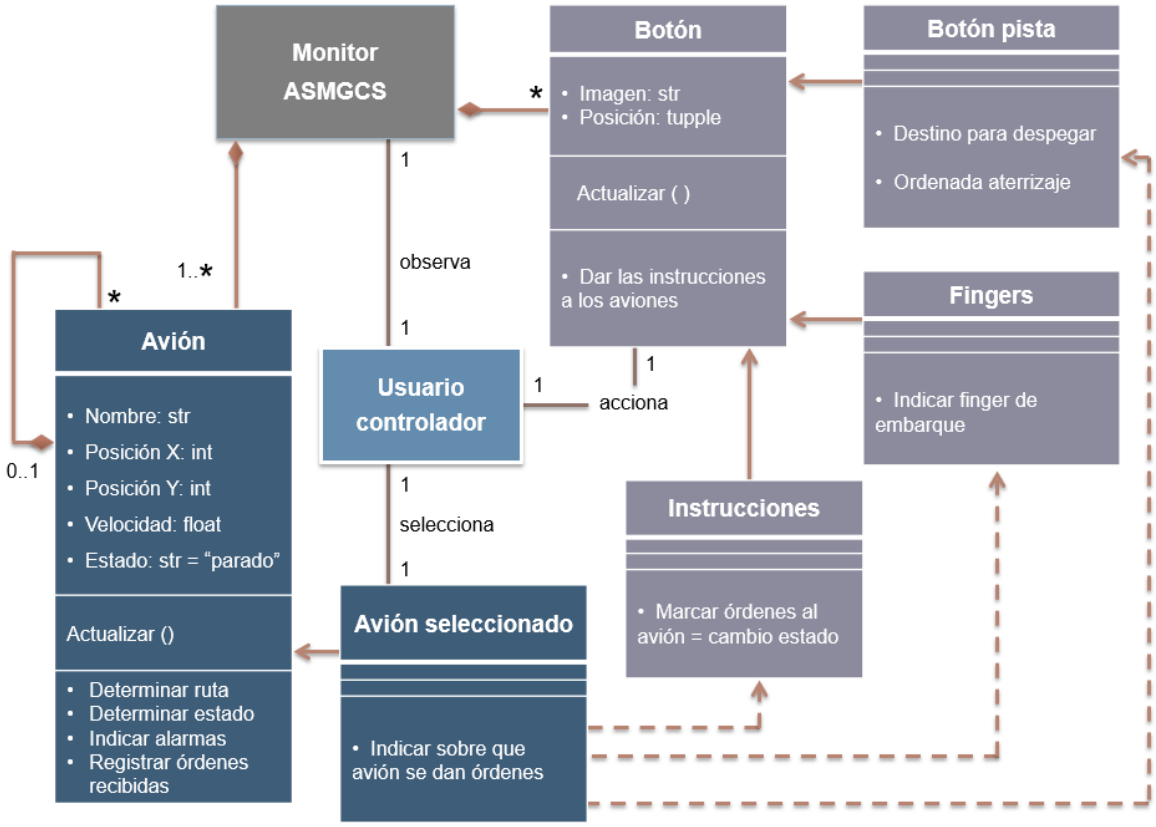
El servicio de control SMGCS puede asignarse a uno o más operadores del ATC equipo.

9. Programa

El conjunto de restricciones, normas y adaptaciones que se realizan en este trabajo se rigen por las normas y métodos recomendados internacionalmente respecto al diseño y ejecución de operaciones en los aeródromos [10].

Por lo tanto, todas las restricciones explicadas en los siguientes puntos del trabajo tratan de acercar al máximo el comportamiento de los elementos de la simulación a la realidad.

9.1. Diagrama de clases UML



9.2. Versión final

En este apartado se explica todo lo que está relacionado con la última versión del programa, desde la composición y aspecto de la interfaz gráfica, hasta los algoritmos desarrollados e implementados.

9.2.1. Interfaz gráfica

Para tener una primera toma de contacto con lo que será el programa realizado se presenta en primer lugar la pantalla que ve el operador.

Se puede ver la configuración de la pantalla en la siguiente figura 9.1:

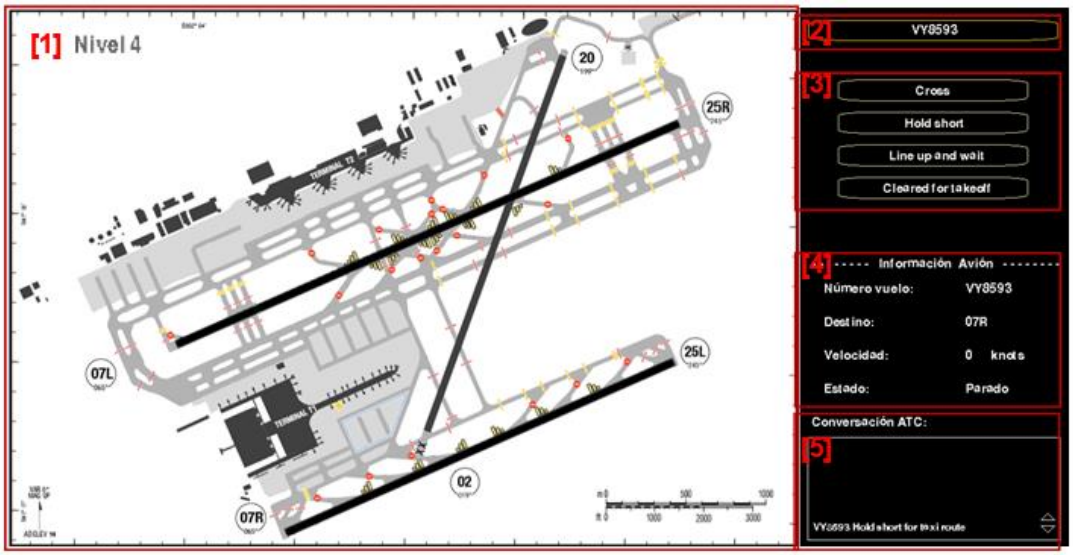


Figura 9.1 Interfaz gráfica sistema control

A continuación, se detallan cada uno de los elementos que componen la pantalla del controlador.

9.2.1.1. Mapa aeropuerto Barcelona y nivel A-SMGCS [1]

Zona de la pantalla donde se produce la simulación del movimiento de los aviones. La profundidad de cálculo y decisión en elección de siguientes rutas depende del nivel de A-SMGCS seleccionado.

9.2.1.2. Número del avión [2]

Panel informativo que indica el número del avión sobre el que se está actuando, para darle órdenes o simplemente ver la información de sus movimientos.

9.2.1.3. Botonera de órdenes [3]

Zona con las cuatro órdenes “principales” que un controlador le puede indicar a un piloto durante su maniobra de rodaje.

9.2.1.4. Información del avión [4]

Muestra la información vinculada al avión seleccionado: su destino, velocidad actual y el estado.

9.2.1.5. Conversación con ATC [5]

Registro de todas las indicaciones que la torre de control ha hecho al piloto durante su maniobra de rodaje.

9.3. Desarrollo programa

En este punto del trabajo se explica el código implementado para el desarrollo correcto del programa, así como las aproximaciones que se han realizado para que el programa simule de la manera más aproximada la realidad.

El programa contempla la vida de los aviones en la superficie del aeropuerto, es decir, las aeronaves que se encuentran en alguna de las fases de *plataforma, rodaje, despegue o aterrizaje.*

Se explican las partes en el orden en que se han implementado, es decir, valores **constantes**, **funciones**, **clases** y **programa principal.** En este apartado solo se presentan las partes de código que se consideran relevantes. Se puede ver el detalle del código implementado en el anejo A.

9.3.1. Valores constantes

En esta parte del programa se incluyen todas aquellas variables que siempre mantendrán el mismo valor.

Los datos del programa que se rigen por esta peculiaridad son:

9.3.1.1. Compañías aéreas

Para acercar la simulación a la realidad se ha incluido una lista con los códigos IATA de las diferentes compañías aéreas que operan actualmente en la terminal T1 del aeropuerto de Barcelona, de este modo se otorgan nombres realistas a los aviones.

Las siguientes variables se han expresado como coordenadas (abscisa y ordenada). Las coordenadas son absolutas respecto al (0,0) que corresponde a la esquina superior izquierda de la pantalla de control.

9.3.1.2. Fingers

Indican las coordenadas de las pasarelas de acceso donde debe el avión recoger o dejar a los pasajeros.

9.3.1.3. Sfinger

Indica el punto correspondiente de las carreteras relacionado con cada finger. El piloto deberá abandonar o incorporarse a la carretera en ese punto, en su maniobra hacia o desde el finger.

9.3.1.4. Destino_auxiliar

Coordenadas de soporte que ayudan al buen funcionamiento de las trayectorias de los aviones. En el punto 9.3.1 del trabajo se puede ver la explicación más detallada respecto a la necesidad de su existencia.

9.3.1.5. Carreteras

Conjunto de listas formadas por dos o más puntos que determinan todas las posibles trayectorias de los aviones. Dichas trayectorias, para simplificar el trabajo, siempre serán rectas y por tanto la definición de una carretera solo tendrá los puntos de inicio y fin estrictamente necesarios. Todos los puntos intermedios los calcula el programa mediante la ecuación de la recta. Se han clasificado en tres tipos de vías como se puede ver en la figura 9.2, de este modo se aproxima el comportamiento de los aviones al real.

- *Carreteras de acceso a los fingers*, marcadas en verde en la figura 9.2, solo se podrán usar cuando el avión se dirija a uno de ellos o haya partido de uno de los mismos.

  En caso de partir de un finger se deberá abandonar este tipo de vías lo antes posible (se detalla en el apartado 9.3.1).

- *Carreteras de salida de las pistas de aterrizaje*, marcadas en amarillo, son de uso exclusivo para los aviones que acaban de aterrizar y desean evacuar la pista. Su uso a la inversa se señaliza en el mapa mediante una señal de prohibido.

- *El resto de carreteras* que intervienen en el funcionamiento de los aviones con terminal asignada la T1, se han señalado en rojo, incluyendo las pistas más habituales de aterrizaje (07L-25R) y despegue (07R-25L).

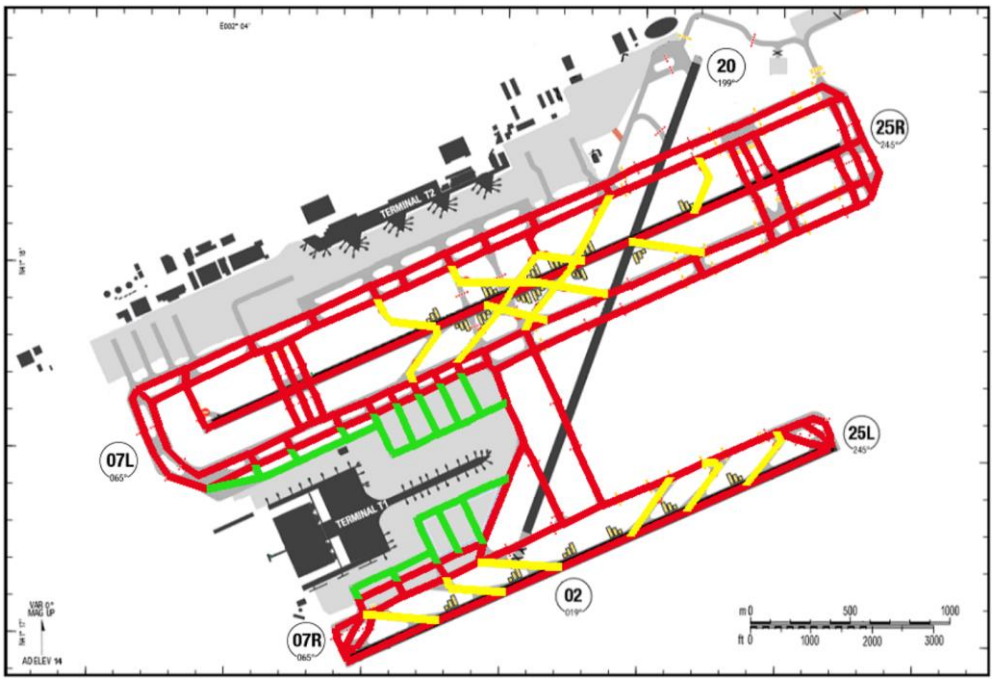


Figura 9.2 Diagrama Aeropuerto Barcelona. Tipología de carreteras

9.3.1.6. Puntos de parada obligada

Las zonas más conflictivas y peligrosas de las vías de rodaje, en general por intersección de alguna pista, se controlan mediante la obligación de parada del avión y posterior autorización de avance desde la torre de control. Se señaliza en el mapa mediante una línea de puntos rojos. Existen dos tipos:

- *Parada obligada bidireccional*: se debe parar en la señalización independientemente de la dirección del avión.

- *Parada obligada unidireccional*: según el sentido de la marcha del avión la zona a la que se dirige es conflictiva o no, y, por tanto, solo se deberá tener en cuenta dicha parada si el siguiente elemento que se encuentra en su camino es la intersección de carreteras.

9.3.2. Funciones

Únicamente se define la función *cargar_imagen* que nos facilita la introducción de

imágenes con y sin transparencia.

9.3.3. Clases

Para manejar los múltiples atributos que deben tener algunos de los componentes del trabajo se han generado un conjunto de clases detalladas a continuación.

9.3.3.1. Cursor

Define el ratón como un objeto. Nos servirá para tratar las colisiones que el usuario desee.

9.3.3.2. Botón

Define las imágenes como objetos, permitiendo así el estudio de colisiones y la actualización de la imagen del botón según el estado (si el curso colisiona con él o no).

9.3.3.3. Avión

Determina los atributos de cada avión y la actualización de ellos. Como se trata de la clase principal se detalla a continuación.

9.3.3.3.1 Atributos

Características que definen todos los parámetros del avión.

- *Nombre del avión*: define el avión, sólo se le da un uso estético para aproximar la plataforma a la realidad.
- *Imagen*: expresa el estado del avión. Si se trata del avión seleccionado se le da otro aspecto.
- *Localización*: se define cada avión como un *rect* y por tanto podremos conocer en todo momento la posición de éste mediante las coordenadas del centro de la imagen.
- *Speed*: velocidad a la que avanza el avión. Se representa en [*knots*] que equivalen a 1.852 km/h.
- *Ruta*: carretera actual en la que se encuentra el avión y que indica los siguientes puntos que debe alcanzar.
- *Función de la recta*: se determina la trayectoria entre puntos de la ruta

mediante la definición de la recta. Para ello se debe determinar la pendiente (*m*), también utilizada para determinar la orientación de la imagen del avión, y el valor independiente de esta (*b*).

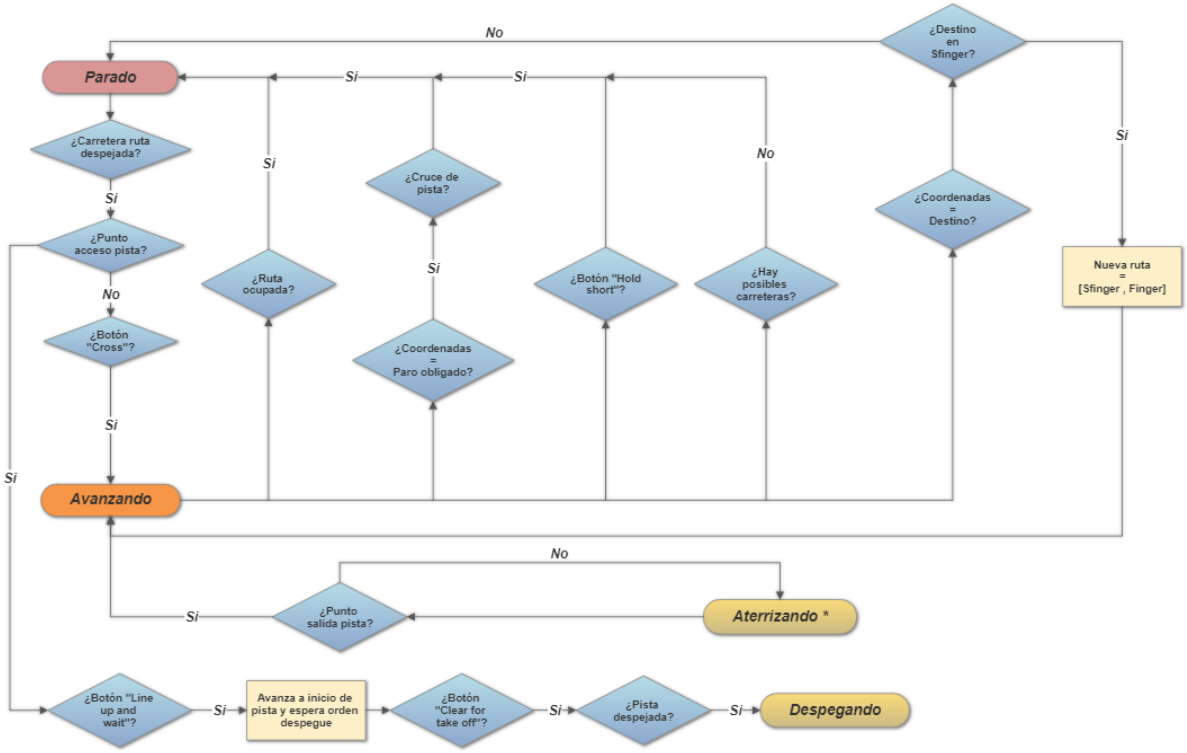
- *Instrucciones*: registro de las órdenes que la torre de control ha indicado al avión durante toda su maniobra de rodaje. También se incluyen las órdenes automáticas.

- *Zona aparcamiento*: las carreteras de acceso a los *fingers (*carreteras verdes de la figura 9.2) solo podrán ser usadas por aquellas aeronaves que tengan la zona de aparcamiento activada, es decir, las que vayan a realizar carga o descarga de pasajeros.

- *Salida de pista*: indica si el avión acaba de aterrizar. En caso afirmativo se le autoriza el uso de las carreteras de salida de pista. De este modo se impide que aviones que estén desplazándose por las vías de taxi invadan una pista.

- *Estado*: determina el comportamiento actual del avión. Se clasifican en 4 posibles estados. Se adjunta un diagrama de estados para entender el comportamiento de dicho atributo (los caminos negativos de la mayoría de puntos de decisión no se representan pues suponen mantener el estado actual):

    - *Avanzando*
    - *Parado*
    - *Despegando*
    - *Aterrizando*



***** *Se detalla el estado "Aterrizando" en el siguiente apartado*

Diagrama 9.1 Flujo de estados avión

9.3.3.3.2 Algoritmos actualización

Según el estado del avión se aplican unos algoritmos u otros, por lo tanto, se clasifican con la misma estructura:

- *Parado*: en este caso la velocidad de avance es cero y por tanto el algoritmo solo procurará que el avión mantenga la configuración de orientación que se ha dado en el último movimiento de avance.

  Solo puede cambiar de estado a "Avanzando" con la botonera de órdenes. Las restricciones que impiden el avance se contemplan en el programa principal (deducibles del diagrama 9.1) y no permitirán el uso del botón "Cross" (se detallan en la configuración de los botones del apartado 9.2.4.1).

- *Avanzando*: el atributo velocidad debe cumplir la normativa ICAO (*International Civil Aviation Organisation).* Para ello se le van aplicando pequeños ajustes de incremento o disminución en función del valor actual, siempre dentro de los límites permitidos.

El algoritmo de avance consiste en el cálculo de rectas entre puntos colindantes de la ruta. Se recuerda que la ruta es la carretera actual que está acercando el avión a su destino. Al no determinar la ruta completa desde el finger hasta la pista o viceversa, se le da cierta flexibilidad al programa para adaptaciones en caso de encontrarse con puntos conflictivos, o con otros aviones en el camino. Una vez se llega al final de la ruta se debe conocer cuál es la siguiente carretera que se abordará y es en este punto cuando se aplica el **algoritmo de cálculo de rutas**:

- Se listan todas las posibles carreteras (restricción de uso según algunos atributos de la aeronave, detallado en el punto 9.2.1.5 Carreteras).

- Se contemplan solo las carreteras que tienen en uno de sus extremos el último punto de la ruta anterior, es decir, que tienen una intersección en común.

- Se ordena esta lista de posibles carreteras según la distancia de su punto final (extremo contrario a la intersección) respecto al destino de la aeronave. Dicha distancia se calcula como la recta que une los dos puntos.

  En caso de haber partido de un finger, como se ha indicado en el apartado *carreteras de acceso a fingers* del punto 9.2.1.5 Carreteras, se debe abandonar dicha zona con la mayor celeridad posible. Para priorizar las carreteras que ayudan a que esto ocurra, se determina su valor de distancia como la inversa negativa de su valor real. Al aplicar el valor negativo se consigue priorizar las carreteras que abandonan la zona de rampas sobre las que no. Se realiza la inversa de este valor para seguir con el criterio de minimizar la distancia al destino, es decir, prioridad de elección de carretera que se aproxima más al destino.

- Se pueden dar dos escenarios según el nivel de A-SMGCS seleccionado:

- **Nivel A-SMGCS 4**: Una vez conocido el orden de preferencia de las candidatas a ser siguiente ruta, se mira, siguiendo el orden de preferencia establecido anteriormente si hay otro avión usándola. En

caso negativo se establece la carretera como *ruta* del avión. En caso contrario se debe mirar la siguiente candidata y así sucesivamente. Puede ocurrir que todas las posibles carreteras estén siendo usadas. En este escenario, el avión cambiaría de estado automáticamente a "Parado" y esperaría a que alguna de las vías se despejase.

Nivel A-SMGCS 2: Este nivel solo se aplica para conocer las rutas óptimas de los aviones. El programa no es tan capaz de tomar decisiones y recalcular rutas. Una vez conocido el orden de preferencia de las candidatas a ser siguiente ruta, se mira si la ruta preferente (la primera) tiene a otro avión usándola. En caso negativo se establece la carretera como *ruta* del avión. En caso contrario el estado del avión pasa automáticamente a "Parado".

Otro algoritmo aplicado al avión en estado "Avanzando" trata de **evitar las colisiones**. Este procedimiento usa parte del algoritmo de cálculo de rutas, en concreto realiza los pasos 1,2 y 4 (de manera específica). Avanza la aplicación de dicho algoritmo a cuando la aeronave se está aproximando a una intersección. En este algoritmo el punto 4 se adapta ligeramente. Se deben consultar tres condiciones concretas:

- ¿Hay alguna carretera de todas las posibles candidatas en uso?
- ¿El avión que la está usando tiene el punto de la intersección como punto final de la ruta?
- ¿Dicho avión se encuentra en estado "Avanzando"?

En caso de que alguna de las condiciones sea negativa se procede con los algoritmos de avance normales. En caso contrario, que se den las tres condiciones, el avión pasará automáticamente a estado "Parado" cediendo así el paso a la otra aeronave.

- *Aterrizando*: es un subestado de "Avanzando" y por tanto se comporta de manera muy similar. El cálculo de trayectorias es idéntico al usado en el estado "Avanzando" (cálculo de rectas entre puntos colindantes de la ruta).

  El atributo velocidad va decreciendo a medida que el avión avanza. Debido a que las salidas de pista son rápidas (ver detalle en el punto 9.2.1.5) no es necesario que

la aeronave llegue a la velocidad de crucero en la pista. El algoritmo de cálculo de siguiente ruta mientras no se alcance el punto de salida se encuentra explicado íntegramente en el estado “Avanzando” (punto anterior del apartado).

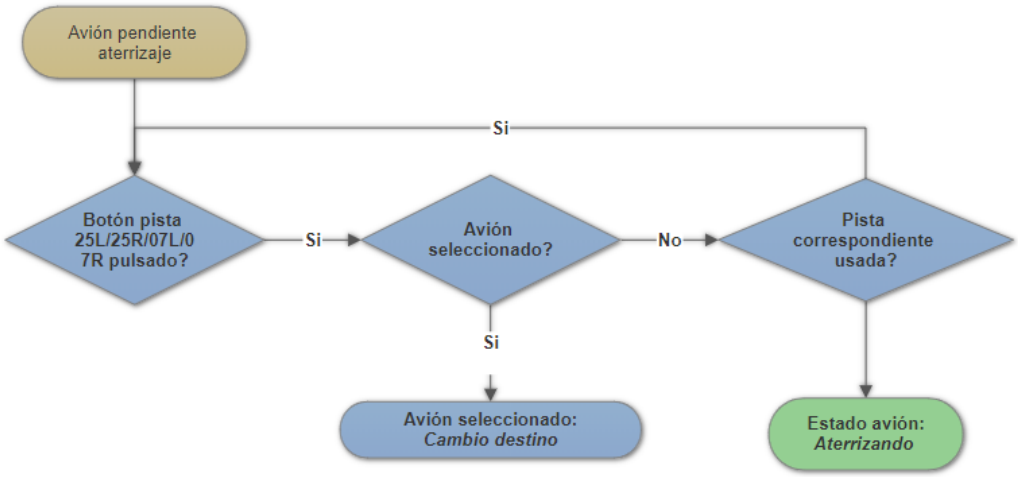


Diagrama 9.2 Detalle estado aterrizaje

- *Despegando*: es un subestado de “Avanzando” y por tanto se comporta de manera muy similar. El cálculo de trayectorias es idéntico al usado en el estado “Avanzando” (cálculo de rectas entre puntos colindantes de la ruta).

  El atributo velocidad va creciendo a medida que el avión avanza. En este caso particular no se debe calcular la siguiente ruta una vez se llega al destino pues simula que el avión ha despegado y ya no es materia de estudio del trabajo, es por ello que una vez se alcanza el destino, el avión desaparece del mapa y de la base de datos (se aplica la función *kill*).

9.3.4. Programa principal

En primer lugar, se cargan todas las imágenes y se generan las listas auxiliares que ayudan al buen desarrollo del programa.

A continuación, se entra en el bucle del programa donde está detallado el algoritmo principal.

En este apartado se explican las funciones y las diferentes llamadas que hace el programa, sin detallar el uso por parte del usuario (el manual de funcionamiento está detallado en el anejo B).

9.3.4.1. Actualización de imágenes

Hay 3 tipos de imágenes:

1. *Pistas de aterrizaje/despegue*: Cambian de aspecto según si se están usando o no. Para verificar su uso se contempla el atributo *Estado* de los aviones. Para conocer que pista se está usando concretamente se mira la posición en la ordenada del avión.

2. *Aspecto de los aviones*: Se actualiza en función de su *Estado*.

   Se añade un aspecto diferente para el avión que se encuentra seleccionado, independientemente de su estado. De este modo se sabe perfectamente sobre que avión se están realizando indicaciones.

3. *Botones*: Se actualizan en función de la colisión con el cursor.

   A continuación, se especifica el funcionamiento:

   a. *Botones de pista*: conjunto de botones formados por 25R/07L/25L/07R.

      La actualización de la imagen solo tiene relación con la colisión.

      Su comportamiento es diferente según si se tiene algún avión seleccionado o no (detallado en el anejo B).

   b. *Fingers*: conjunto de botones situados en las coordenadas de los *fingers*.

      La actualización de la imagen solo tiene relación con la colisión.

      Su comportamiento es diferente según si se tiene algún avión seleccionado o no (detallado en el anejo B).

   c. *Botones órdenes ATC*: conjunto de botones con las órdenes "*Cross", "Hold short", "Line up and wait", "Clear for take off".*

      Solo actúan en caso de tener un avión seleccionado.

      La actualización de la imagen tiene relación con la colisión y a su vez con el estado y destino del avión seleccionado. En caso de que la orden no pueda ser dada en cierta configuración el botón aparecerá bloqueado en un color granate. Las restricciones se pueden deducir del diagrama de flujo

simplificado del apartado 9.*2.3.3. Avión – Atributos – Estado*, puesto que solo se incluyen los caminos afirmativos:

- "*Cross*"

  1. Condiciones para que el botón esté activado son:

     - El estado del avión seleccionado debe ser "*Parado".*
     - Si nos encontramos en un *finger* se debe tener como atributo *destino* alguna de las pistas.
     - Las coordenadas del avión no deben coincidir con las de un punto de acceso a pista.

  2. Condiciones para que el botón actúe:

     - Carretera de la ruta no está siendo usada por otro avión.

- "*Hold short*"

  1. Condiciones para que el botón esté activado son:

     - El estado del avión seleccionado debe ser "*Avanzando".*
     - La ruta del avión no deben ser una carretera de acceso a pista.

  2. Condiciones para que el botón actúe:

     - Siempre que esté activado debe actuar si el controlador lo considera.

- "*Line up and wait*"

  1. Condiciones para que el botón esté activado son:

     - El estado del avión seleccionado debe ser "*Parado".*
     - Las coordenadas del avión deben coincidir con las de un punto de acceso a pista.

2. Condiciones para que el botón actúe:

    - Siempre que esté activado debe actuar si el controlador lo considera.

- "*Clear for take off*"

    1. Condiciones para que el botón esté activado son:

        - El estado del avión seleccionado debe ser "*Parado".*
        - Las coordenadas del avión deben coincidir con las de un punto de inicio de pista.
        - La pista correspondiente tiene que encontrarse vacia.

    2. Condiciones para que el botón actúe:

        - Siempre que esté activado debe actuar si el controlador lo considera.

9.3.4.2. Actualización de textos

Todos los detalles escritos en la *Información Avión* se actualizarán en función del avión que se encuentre seleccionado.

9.3.4.3. Actualización de aviones

La generación de aviones se realiza mediante los botones *finger* o *botones de pista*. Solo se podrá "generar" (permitir que un avión vaya desde el aparcamiento a un *finger* o permitir que un avión aterrice) un avión nuevo si hay *fingers* disponibles.

En general la actualización de los aviones una vez se encuentran en el aeropuerto es automática y en función de los algoritmos, que a su vez dependen de los *Estados*, explicados en el punto 9*.2.3.3. Aviones – Algoritmos actualización*.

Solo se podrán/deberán realizar modificaciones sobre el avión seleccionado:

- Modificar destino
- Modificar estado mediante los botones activados

9.3.4.4. Actualización conversación ATC

El conjunto de órdenes que ha recibido un avión se almacena en el atributo *instrucciones*. Cada nueva orden añade automáticamente el lenguaje utilizado adecuado a dicha indicación.

Se puede ver el conjunto de instrucciones en la parte inferior derecha de la pantalla. Mediante el *scroll* podemos acceder a instrucciones más antiguas.

9.4. Problemas principales

En este apartado se explican las principales dificultades que se han presentado durante la implementación del programa y como se ha tratado de solucionarlas.

9.4.1. Movimiento

El sistema de control de posición de los *sprites* se ha realizado mirando las coordenadas del centro de la imagen. En pygame estas coordenadas se representan como número de píxeles respecto a la esquina superior izquierda, donde se haya el (0,0).

Los píxeles se tratan como números enteros, esto ha generado dos problemas:

1. El avance no es suave y rectilíneo:

   Por mucho que la ecuación de la recta aplicada en el algoritmo de avance de los aviones calcule el punto exacto a donde debe avanzar el avión, el programa siempre redondeará al píxel más cercano (números enteros).

   No se encontró una mejor solución.

2. Problemas para detectar el destino:

   En ciertas situaciones y al redondear las coordenadas de manera automática, no se detectaba exactamente el punto de destino como coordenadas actuales del avión, y el avión se colapsaba cerca del destino hasta que coincidía.

   Para solucionar este problema se consideran punto de destino todas las coordenadas que estén dentro del círculo que tiene como centro el punto de

destino y un radio de 3 píxeles. La solución no es relevante en el comportamiento de los aviones, pero visualmente se consigue un movimiento más suavizado.

En las figuras siguientes se muestra el despegue desde 07R:

- Antes de aplicar la solución, el avión se queda colapsado en el punto de destino hasta que encuentra las coordenadas exactas, aparentemente el avión va cambiando de sentido bruscamente.

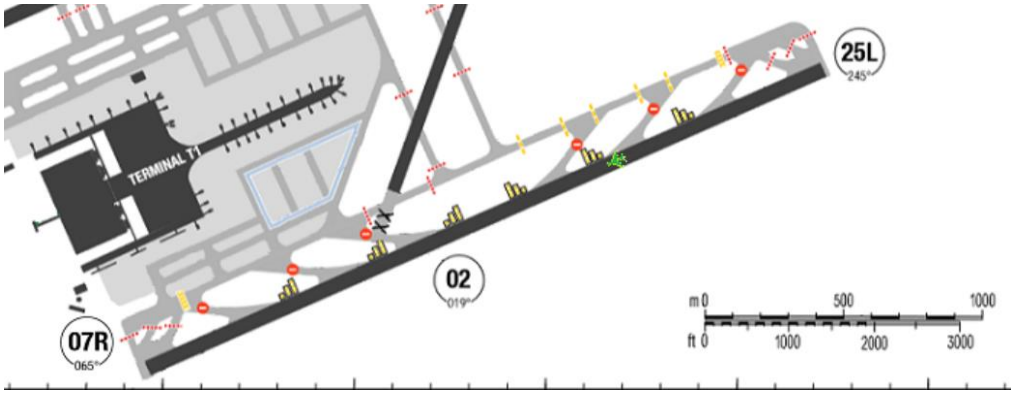


Figura 9.3 Detalle fallo detección coordenadas destino

- Después de aplicar la solución, cuando el centro del avión entre dentro del círculo rojo el programa interpretará que está situado en su destino:

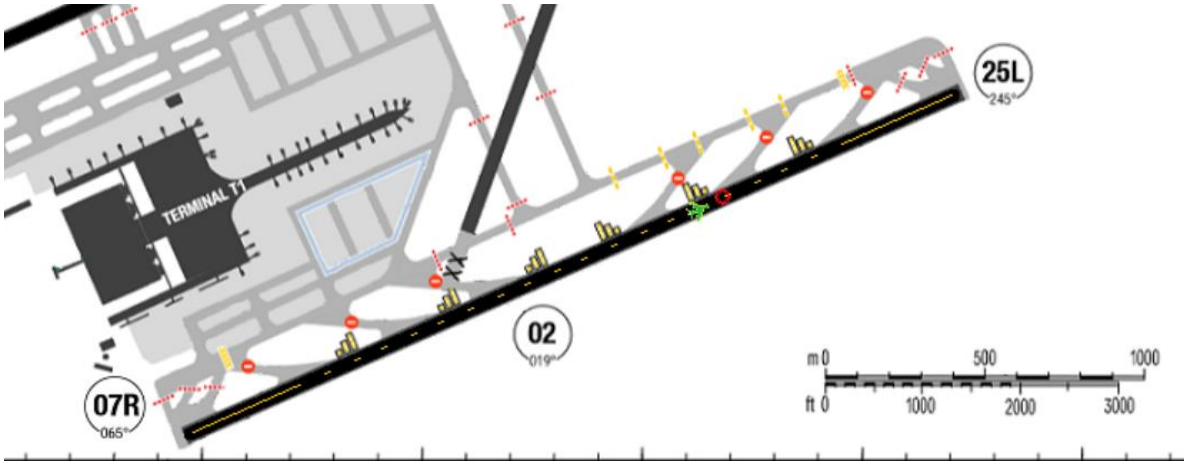


Figura 9.4 Detalle solución detección coordenadas destino

Se producía otro problema de movimiento, ya no relacionado con el trato de los píxeles como números enteros.

3. Parón de los aviones en las intersecciones:

Inicialmente se esperaba hasta llegar al final de una ruta para calcular la siguiente, esto provocaba que en las intersecciones los aviones realizaran una pequeña parada mientras se calculaba dicha ruta.

Para solucionarlo se aplicó un método similar al de cálculo de destino del apartado anterior. Se adelantó el cálculo de la siguiente ruta al momento en el que el avión se encuentra a una distancia de 3 píxeles o menos de la intersección, consiguiendo así un movimiento suavizado y sin parones.

4. Destino auxiliar

   El algoritmo aplicado al cálculo de rutas, detallado en el apartado 9.2.3.3.2, que busca minimizar la distancia entre el punto final de la posible ruta y el destino generaba errores en ciertos casos.

   Estos errores se daban en dos escenarios muy concretos:

   - El avión se situa en la zona de *fingers* superior y tiene como destino el punto de entrada a pista 07R.
   - El avión se situa en la zona de *fingers* inferior y tiene como destino el punto de entrada a pista 07L.

Cuando se daba una de estas dos situaciones el avión se quedaba colapsado en un bucle y no conseguía llegar al destino. Esto se producía debido a que para poder llegar a carreteras que accediesen al destino primero debía tomar rutas que se alejaban de este, concepto contrario al funcionamiento implementado con el algoritmo de cálculo de rutas. En la figura 9.5, a continuación, se muestra un ejemplo de lo explicado:

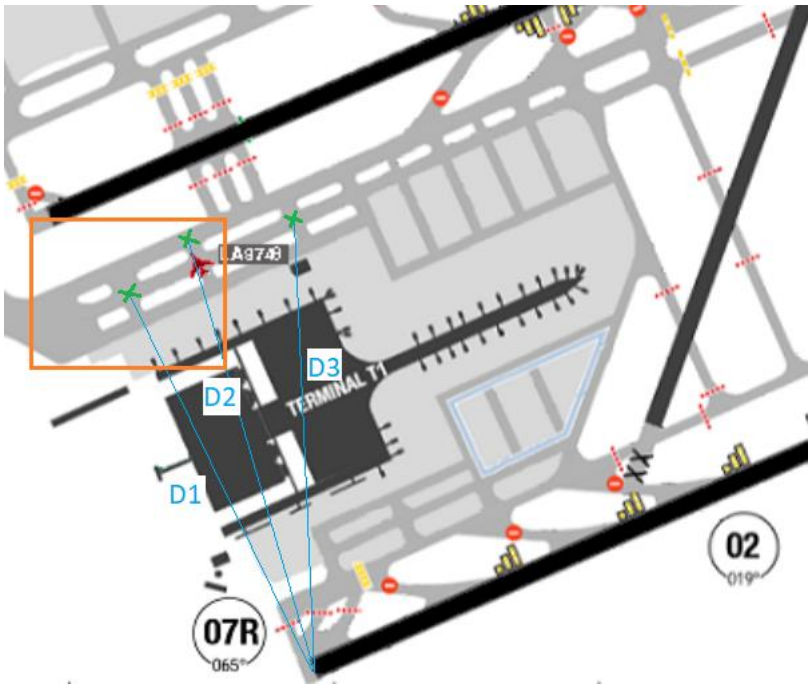


Figura 9.5 Detalle fallo algoritmo cálculo de rutas

Las cruces simbolizan los tres posibles finales de la siguiente ruta. Quedan por tanto definidas tres posibles distancias al punto de destino: D1, D2 y D3. Dichas distancias tienen un valor aproximado respectivamente de 207, 222 y 222.

Por tanto, el avión se dirigirá al punto de la D1 y se quedará realizando todo el rato la ruta dentro de la zona señalada en naranja, en lugar de bordear la terminal 1 y poder alcanzar su destino.

Para solucionar este problema, en caso de encontrarnos con uno de estos escenarios, se establece un destino auxiliar intermedio que se debe alcanzar previamente a dirigirse al destino final. En la siguiente figura 9.6 se marcan los destinos auxiliares, en verde el usado en el primer escenario y en naranja el segundo:

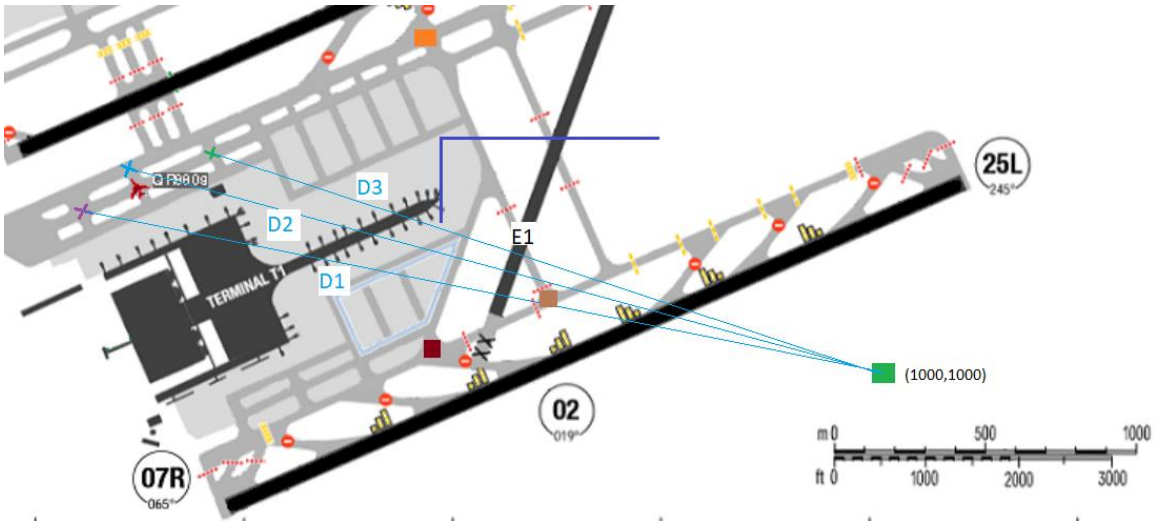


Figura 9.6 Detalle solución algoritmo cálculo de rutas

Con esta nueva metodología las distancias de las posibles rutas al destino quedan redefinidas como se puede ver en la figura 9.6, y sus nuevos valores son aproximadamente 940, 934 y 899, respectivamente. Por lo tanto, se toma la ruta hacia la cruz verde y de esta manera se rodeará la T1 dando el acceso necesario a las carreteras en contacto al destino final. Como el destino auxiliar del escenario 1 (cuadrado verde) se encuentra fuera de las carreteras una vez superada la línea azul se restablece su destino original. En el segundo caso (cuadrado naranja) se restablece el destino una vez se alcance este destino auxiliar.

Se tuvo que establecer un punto de destino auxiliar fuera de las carreteras para no limitar las posibles rutas. Previamente se habían probado los destinos auxiliares de color granate y marrón:

- Destino auxiliar granate: no solucionaba el problema, seguía sucediendo lo mismo.
- Destino auxiliar marrón: fuerza el paso por la carretera E1.

5. Colapso zona aparcamiento

   En primera instancia, teniendo en cuenta las tipologías de las carreteras tal y como se ve en la figura 9.2, se permitía la circulación por las carreteras de acceso a los *fingers* mientras se tuviera el permiso sin más restricciones. Este hecho provoca que en los casos en los cuales el destino se encuentra en el lado opuesto de la T1, el avión permanecerá en las carreteras de acceso a los *fingers* durante toda la acción de rodeo de la T1.

   Debido al número elevado de aviones en la zona de *fingers* y con el

comportamiento de los aviones expuesto en el párrafo anterior, se producían colapsos muy importantes.

Para solucionar este problema, y de este modo acercar las rutas simuladas a las reales, se le da prioridad a las vías que llevan al avión fuera del acceso a *fingers* (siempre que el destino no sea uno de estos).

La solución aplicada se explica en el punto 9.2.3.3.2. A continuación se adjunta una imagen para explicarlo gráficamente:

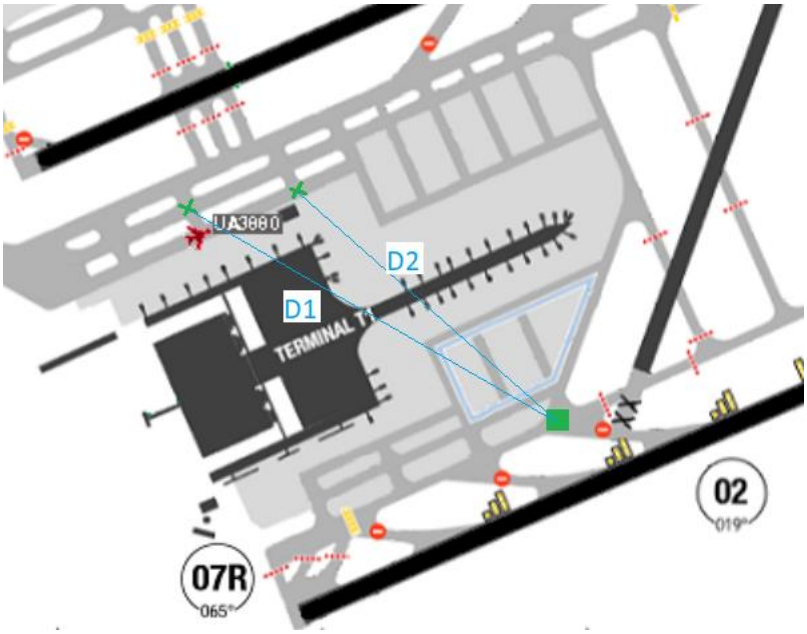


Figura 9.7 Detalle colapso zona acceso fingers

Antes de aplicar la solución, el avión en cuestión se dirigiría hacia el punto final de D2 pues la distancia es menor al destino auxiliar que si va hacia el extremo de la D1. De esta manera esta aeronave se mantiene en la zona de aparcamiento, zona con mucho tráfico.

Una vez aplicada la solución se dirigiría hace el extremo de D1 ya que -1/D1 es más pequeño que D2. De esta manera queda despejada la zona de acceso a *fingers* y se evitan colapsos y ajustes de ruta en esta zona para evitar otras aeronaves.

6. Orientación aviones

Debido al uso de coordenadas absolutas en el posicionamiento de los elementos, se marcará la orientación de los aviones girando la imagen de los aviones el número de grados que corresponda, en lugar de dar las coordenadas del extremo

delantero.

9.4.2. Colisiones

Los algoritmos de cálculo de ruta que se empezaron aplicando, miraban de entre las posibles carreteras que partían de una intersección cuales no estaban utilizadas. De esta manera se conseguía que dos aviones no estuviesen nunca usando la misma carretera.

Este procedimiento contenía un error. Se podían dar dos casos:

1. La intersección solo tenía dos carreteras. Una de ellas tenía que ser la usada actualmente y si la otra estaba ocupada el avión se paraba y no se producía la colisión. En este caso funcionaba bien.

2. La intersección tenía más de 2 carreteras. Una de ellas ocupada por el avión estudiado, otra de ellas ocupada por otro avión y aún quedaban posibles carreteras disponibles. Entonces podía darse que se encontrasen en la intersección y colisionasen.

   Para solucionarlo se aplicaron las tres condiciones explicadas en el apartado *9.2.3.3.2 Algoritmos actualización – Avanzando.* Si alguna de todas las carreteras que tienen como punto en común la intersección está ocupada por un avión en movimiento hacía la intersección, el avión estudiado cambiará su *estado* a "*Parado*" evitando así la colisión.

10. Planificación y programación

Una vez realizado este proyecto se debe proceder con su implantación en la realidad.

En primer lugar, deberíamos realizar un conjunto de tests para verificar el buen funcionamiento del sistema y comprobar que es capaz de aportar las funciones requeridas por un A-SMGCS de nivel 2.

Se analizan los *stakeholders* del proyecto: compañías aéreas, AENA, aseguradoras, fabricantes de los sistemas de control, compañías tecnológicas…

También se deberán buscar inversores potenciales para conseguir financiación en todas las fases del proyecto.

Posteriormente se procedería a la implantación del nuevo sistema A-SMGCS de nivel 2, dividida en tres bloques:

- Introducción del sistema (interfaz gráfica) en las tecnologías existentes. En este proyecto como se trata del nivel 2 del sistema se parte de una base de conexiones bien implementada, requerida a la hora de introducir el nivel 1 del sistema.

- Adaptación de las infraestructuras del aeropuerto. Por ejemplo, la instalación de un sistema SMR más potente.

- Adaptación aeronaves y torre de control. Se debe incorporar la interfaz gráfica y los sistemas necesarios para los avisos o alarmas generados por el A-SMGCS.

A lo largo de todo el proceso se debe realizar una exhausta formación de los operadores para el buen funcionamiento del sistema.

Posteriormente se deberá proceder con las operaciones de mantenimiento correspondientes y un proceso de mejora continua de los sistemas de control de superficie del aeropuerto para evolucionar y hacer del transporte aéreo más seguro.

11. Análisis económico [8]

11.1. Coste económico

El aeropuerto de Barcelona actualmente dispone del primer nivel de A-SMGCS, como se puede ver en el apartado 8. Por lo tanto, solo se considerarán como costes del proyecto la diferencia entre el coste de implementación del nivel 2 de A-SMGCS y el sistema actual.

Los costes promedios de implementación del nivel 1 y 2 del A-SMGCS fueron analizados por Eurocontrol. En ellos se incluye la instalación de infraestructuras, los requisitos técnicos para la fusión de datos y la instalación de la interfaz gráfica. No incluyen el coste de instalación de un radar de superficie SMR, se supone ya disponible en el aeropuerto.

Puesto a que el número de operaciones anuales es de aproximadamente 330.000 se considera que el aeropuerto de Barcelona entra en el escenario de aeropuerto grande propuesto por Eurocontrol.

SECTOR	COSTE SISTEMA	COSTE IMPLEMENTACIÓN	COSTE TOTAL	COSTE MANTENIMIENTO ANUAL
ANSP	3.4M€	0.653M€	4.2M€	0.342M€
Aeropuerto	0.3M€	0.038M€	0.338M€	0.038M€

Tabla 11.1 Coste instalación nivel 1 A-SMGCS

De la tabla anterior podemos resumir el coste medio de implementación del nivel 1 en 4.538M€ más un coste de mantenimiento anual de 0.38 M€.

Los costes de implementación del nivel 2 difieren del nivel 1 debido a la necesidad de instalar tecnología adicional al SMR. Esta mejora es necesaria para obtener mas precisión en la detección y evitar falsas alarmas.

A-SMGCS NIVEL 2	COSTE SISTEMA	COSTE IMPLEMENTACIÓN	COSTE TOTAL	COSTE MANTENIMIENTO ANUAL
Aeropuertos grande	1.35M€	0.75M€	2.1M€	0.068M€

Tabla 11.2 Coste instalación nivel 2 A-SMGCS

De la tabla anterior podemos resumir el coste medio de implementación del nivel 2 en 6.638M€ (4.538+2.1) más un coste de mantenimiento anual de 0.448M€ (0.38+0.068).

En conclusión, el coste de este proyecto para mejorar el nivel implementado de A-SMGCS es de 2.1M€ más 0.068 M€ al año.

También se debe contemplar el coste de generación del programa, es decir, el coste de 320 horas que se ha tardado aproximadamente en desarrollar el sistema. El coste total, suponiendo un precio de la hora de 60 €/hora, asciende a 19.200 €.

Por tanto el coste total del proyecto es de 2.1192 M€ más 0.068 M€ al año.

11.2. Beneficio económico

Cuando se habla de los beneficios económicos de este trabajo, nos referimos a los costes que se evitan al prevenir un incidente o accidente de aviación y al reducir los tiempos de rodaje.

11.2.1. Temporal

Los beneficios principales del sistema A-SMGCS derivan de la reducción de los tiempos de rodaje y al aumento de rendimiento en condiciones meteorológicas adversas (baja visibilidad).

En este proyecto no se contemplarán puesto que el beneficio ya se obtiene en el nivel 1. Se deja la tabla de beneficios a continuación de manera informativa:

AIRLINE OPERATORS	€ per annum
LEVEL 1	
Reduction in delays	0.36M
Reduction in taxi duration (all visibilities)	0.28M
TOTAL Level 1	0.64M

Tabla 11.3 Beneficio anual temporal nivel 2 A-SMGCS sobre nivel 1

11.2.2. Seguridad

El valor añadido del nivel 2 sobre el nivel 1 se da en el aspecto de seguridad. Este parámetro contempla el ahorro en costes por reparación de vehículos e infraestructuras dañados en incursiones de la pista. A continuación, se indica el beneficio generado al

implementar el nivel 2:

AIRLINE OPERATORS	€ per annum
LEVEL 2	
Safety	0.26M

Tabla 11.4 Beneficio anual en seguridad nivel 2 A-SMGCS sobre nivel 1

11.2.3. Humano

Se estudia el número de víctimas y heridos graves de los entre los años 2007 y 2016 (10 años). Como se puede ver en la figura 5, en el año 2008 hubo un grave accidente en Madrid que supuso 154 fallecidos.

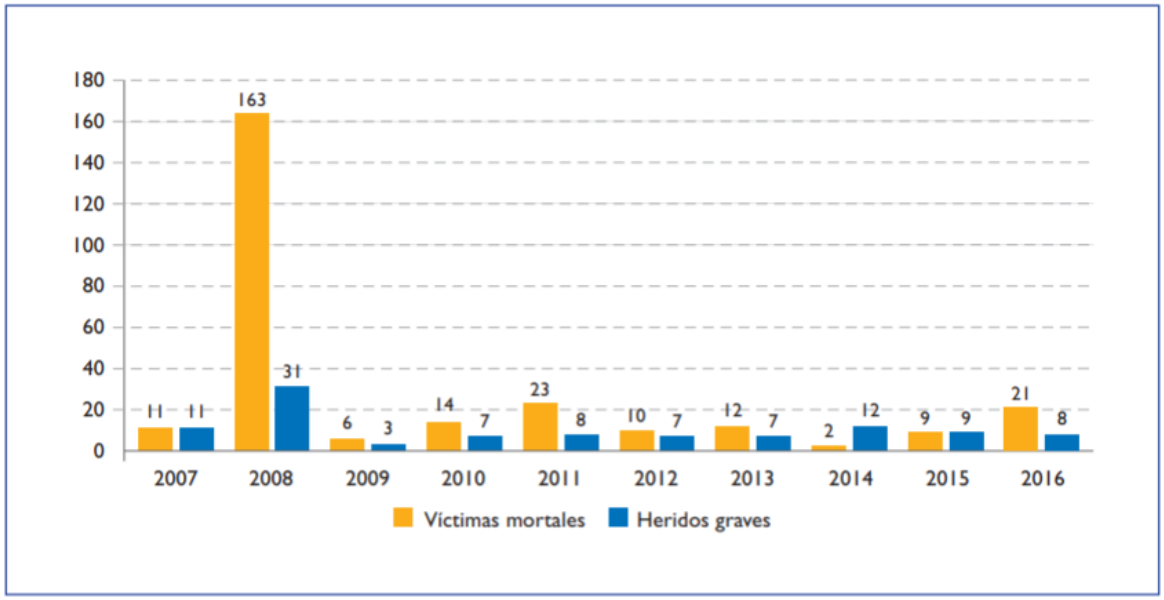


Diagrama 11.1 Evolución temporal número de víctimas y heridos graves en accidentes aéreos

Para que el estudio no se desvirtúe, se decide analizar cada cuanto ocurre un accidente de semejantes dimensiones.

A continuación, se adjunta una tabla con los mayores accidentes de la aviación española:

Tabla 11.5 Accidentes más graves de la historia de la aviación española

Descripción	Año	Nº Fallecidos

Accidente del Comet 4 de Dan Air	1970	112
Vuelo 602 de Iberia	1972	104
Accidente del Convair 990 de Spantax	1972	155
Vuelo 118 de Aviaco	1973	85
Accidente de Los Rodeos	1977	583
Vuelo 1008 de Dan Air	1980	146
Vuelo 995 de Spantax	1982	50
Vuelo 11 de Avianca	1983	181
Colisión vuelos 350 de Iberia y el 134 de Aviaco	1983	93
Vuelo 610 de Iberia	1985	148
Vuelo 4101 de PauknAir	1998	38
Vuelo 5022 de Spanair	2008	154

Diagrama 11.2 Evolución temporal del número de fallecidos en los accidentes más graves de la aviación española

Del diagrama 11.2 se puede extraer que en los últimos años ha disminuido de manera muy notable el número de accidentes graves. También podemos concluir que no se desvirtúa el estudio si contemplamos el accidente de 2008, pues aproximadamente cada 10 años ocurre un accidente similar en España.

ETSEIB

Del diagrama 11.1 se puede extraer que al año en España se producen de media 27 víctimas mortales y 10 heridos graves por accidentes de aviación.

Solo se consideran las víctimas mortales para simplificar los cálculos y estimar un beneficio a la baja.

Según la normativa europea [5] la cobertura mínima de seguro por pasajero que debe tener una compañía aérea es de 250.000 derechos especiales de giro (DEG), o de 100.000 DEG en aeronaves de peso inferior a 2.700 kilos. Actualmente, 1 DEG corresponde a 1,20527 €.

Por tanto, y suponiendo que se trata de aeronaves con peso menor a 2.700 kilos, el coste total mínimo en indemnizaciones asciende a 3,25 millones de euros.

Para encontrar la probabilidad de que esas víctimas se den en el aeropuerto de Barcelona, se considera la proporción de operaciones realizadas en este aeropuerto respecto a las de todo el territorio español. El aeropuerto de Barcelona absorbe aproximadamente el 15% de las operaciones.

Por tanto, de manera estadística podemos concluir que el coste en indemnizaciones por víctimas en el aeropuerto de Barcelona es de 0.484M€ anuales.

11.2.4. Ambiental

Este proyecto genera cierto beneficio ambiental debido a la reducción de emisiones por parte de las aeronaves. Este fenómeno se da gracias a la reducción de tiempos en las maniobras de rodaje.

11.3. Resumen balance económico

Realizando la síntesis de los costes y beneficios se obtiene:

	Año 0	Año 1	Año 2	Año 3	Año 4	Año 5
Costes						
Implementación nivel 2	2,1192					
Mantenimiento		0,068	0,068	0,068	0,068	0,068
Beneficios						
Seguridad		0,26	0,26	0,26	0,26	0,26
Indemnizaciones*		0,3146	0,3146	0,3146	0,3146	0,3146
Resultado	-2,1192	-1,6126	-1,106	-0,5994	-0,0928	0,4138

* Se estima la reducción del 65% de las víctimas (debido al alto ratio víctimas/accidente prevenir un accidente reduce notablemente las víctimas)

La inversión tiene un periodo de retorno de 4 años, a partir del 5 año se empezarían a generar beneficiosos económicos.

Destacar también los beneficios humanos y ambientales que tienen gran importancia y peso a la hora de decidir sobre la implantación de este tipo de sistemas.

Anejo A: Código programa

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# Módulos
import sys,pygame
import random
from pygame.locals import *
from math import *
# Constantes

#--------------------SIGLAS COMPAÑIAS AEREAS----------------------------

CompanyiasaereasT1=
["A3","SVO","AR","AH","BT","AB","AC","CA","UX","AF","YW","AZ","AA","IZ","OS","AV","CJ","BA","SN","OU","CSA","DL","CAI","LY","EK","AY","IB","6H","KL","KE","LA","LO","LH","5L","QR","FV","AT","RJ","SK","SQ","LX","S7","TP","RO","TU","TK","PS","UA","VY"]

#--------------------COORDENADAS CARRETERAS----------------------------
AS=[[243,443],[237,432]]
BS=[[272,415],[277,428]]
CN=[[578,173],[583,184]]
CS=[[319,394],[324,407]]
DS=[[356,394],[350,382]]
D2D1=[[510,332],[533,382],[551,426],[576,481]]
D3=[[502,313],[510,332]]
E2E1=[[500,421],[511,441],[534,491],[538,499]]
E31=[[500,421],[495,406]]
E32=[[495,406],[484,383]]
E4=[[473,351],[484,383]]
E51=[[473,351],[466,345]]
E52=[[466,345],[462,333]]
EN=[[507,203],[512,216]]
ES=[[371,372],[378,384]]
ES1=[[395,424],[378,384]]
FN=[[461,223],[466,236]]
FS=[[394,363],[399,375]]
FS1=[[417,413],[399,375]]
G1=[[743,411],[778,398],[789,401],[794,414],[795,426]]
G10=[[347,586],[355,587],[352,600],[339,619],[327,621],[331,631]]
G11=[[347,586],[337,601],[329,618],[331,631]]
G2=[[743,411],[767,402],[777,408],[789,416],[795,426]]
G3=[[743,411],[763,418],[777,422],[795,426]]
G4=[[714,462],[747,420],[743,411]]
G5=[[657,487],[688,444],[687,441],[685,440],[682,440],[678,440],[672,441]]
G6=[[603,511],[641,455]]
G7=[[457,536],[537,540]]
G8=[[483,564],[429,561],[428,558],[426,551]]
G9=[[347,586],[419,591]]
GATEAS=[[247,455],[243,443]]
GATEBS=[[277,428],[284,439]]
GATECS=[[331,418],[324,407]]
GATEDS=[[361,404],[356,394]]
GATEKS1=[[441,474],[452,469]]
GATEKS2=[[452,469],[462,464]]
GATEKS3=[[462,464],[470,461]]
GATEKS4=[[470,461],[486,455]]
GATEIS=[[484,383],[459,394]]
GATENS=[[412,526],[417,537]]
GATEPS=[[374,544],[379,554]]
GATEQS=[[337,561],[342,571]]
GN=[[425,239],[431,252]]
GS=[[414,354],[420,365]]
GS1=[[438,404],[420,365]]
HN=[[382,259],[387,272]]
HS=[[436,344],[440,355]]
HS1=[[459,394],[440,355]]
```

```
J10=[[342,571],[379,554]]
J5=[[500,421],[486,455]]
J6=[[486,455],[461,518]]
J7=[[461,518],[438,527]]
J8=[[438,527],[417,537]]
J9=[[379,554],[417,537]]
JN=[[354,270],[360,284]]
K10=[[347,586],[357,583],[385,568]]
K11G12=[[347,586],[320,600],[321,608],[331,631]]
K2=[[672,441],[733,415],[743,411]]
K4=[[576,481],[621,463],[641,455]]
K3=[[641,455],[672,441]]
K5=[[538,499],[576,481]]
K6=[[457,536],[485,524],[529,503],[538,499]]
K8=[[426,551],[457,536]]
K9=[[385,568],[426,551]]
KN=[[296,297],[301,309]]
L101=[[417,413],[406,419]]
L102=[[406,419],[395,424]]
L111=[[395,424],[389,426]]
L112=[[389,426],[381,430]]
L113=[[381,430],[375,432]]
L114=[[375,432],[372,428]]
L115=[[372,428],[369,421]]
L116=[[369,421],[361,404]]
L121=[[345,411],[361,404]]
L122=[[331,418],[345,411]]
L131=[[331,418],[312,426]]
L132=[[312,426],[299,432]]
L133=[[299,432],[292,436]]
L134=[[292,436],[284,439]]
L141=[[284,439],[267,447]]
L142=[[267,447],[257,452]]
L143=[[257,452],[247,455]]
L81=[[459,394],[447,401]]
L82=[[447,401],[438,404]]
L91=[[417,413],[425,410]]
L92=[[425,410],[438,404]]
LN=[[237,322],[242,335]]
LS=[[461,518],[457,536]]
LS1=[[441,474],[461,518]]
M1=[[820,194],[832,189],[835,183]]
M1a=[[835,183],[828,170]]
M125R=[[835,183],[841,162],[835,149],[825,130],[812,136]]
S1=[[808,89],[819,114],[825,130],[812,136]]
S2S1=[[787,79],[808,89]]
T2S1=[[795,93],[808,89]]
M10=[[378,384],[399,375]]
M11=[[378,384],[356,394]]
M12=[[356,394],[324,407]]
M13=[[324,407],[277,428]]
M14=[[277,428],[243,443]]
M15=[[243,443],[197,465]]
M16=[[197,465],[159,453],[144,430]]
M15M16L14=[[197,465],[247,455]]
M2=[[762,219],[808,199],[820,194]]
M2N2=[[820,194],[814,175]]
M3N3_1=[[739,230],[735,214]]
M3N3_2=[[762,219],[756,202]]
M31=[[672,260],[739,230]]
M32=[[739,230],[762,219]]
M5M4=[[510,332],[561,309],[596,294],[672,260]]
M6=[[473,351],[510,332]]
M7=[[440,355],[466,345]]
M8=[[440,355],[420,365]]
M9=[[399,375],[420,365]]
MS1=[[418,485],[438,527]]
N1=[[828,170],[820,149],[812,136]]
N10=[[394,363],[371,372]]
```

ETSEIB

```
N11=[[371,372],[350,382]]
N12=[[350,382],[319,394]]
N134=[[319,394],[309,398]]
N133=[[309,398],[297,402]]
N132=[[297,402],[286,408]]
N131=[[286,408],[272,415]]
N14=[[272,415],[237,432]]
N15=[[237,432],[189,452]]
N16=[[189,452],[173,442],[158,427]]
N21=[[756,202],[814,175]]
N22=[[814,175],[828,170]]
N31=[[674,238],[735,214]]
N32=[[735,214],[756,202]]
N4=[[581,278],[603,270],[664,243]]
N5=[[523,303],[568,285],[581,278]]
N6=[[462,333],[502,313]]
N6N5=[[502,313],[523,303]]
N7=[[462,333],[436,344]]
N8=[[436,344],[414,354]]
N9=[[414,354],[394,363]]
NM=[[664,243],[672,260]]
NS=[[426,551],[417,537]]
P1=[[664,201],[678,167],[668,148]]
P2=[[546,253],[583,184]]
P3=[[559,246],[513,241]]
P4=[[486,280],[513,241]]
P5=[[512,267],[437,267],[431,252]]
P6=[[416,309],[373,305],[366,297],[360,284]]
EnlaceP3P4=[[513,241],[506,226],[512,216]]
PISTA1=[[202,404],[602,228],[664,201],[799,143]]
PISTA2=[[]]
PISTA3=[[331,631],[419,591],[483,564],[537,540],[657,487],[795,426]]
PN=[[176,366],[176,350]]
PS=[[379,554],[385,568]]
Q101=[[337,561],[357,551]]
Q102=[[357,551],[374,544]]
Q71=[[418,485],[426,481]]
Q72=[[426,481],[434,478]]
Q73=[[434,478],[441,474]]
Q81=[[412,526],[402,505]]
Q82=[[402,505],[400,501]]
Q83=[[400,501],[398,494]]
Q84=[[398,494],[403,492]]
Q85=[[403,492],[409,489]]
Q86=[[409,489],[418,485]]
Q9=[[374,544],[412,526]]
QS=[[342,571],[347,586]]
RN=[[129,370],[143,382]]
N3N4=[[674,238],[664,243]]
R1=[[602,228],[674,238]]
R2=[[512,267],[581,278]]
R3=[[546,253],[506,302],[502,313]]
R4=[[463,289],[523,303]]
R5=[[486,280],[473,299],[436,344]]
R6=[[416,309],[417,317],[391,354],[394,363]]
S10=[[296,297],[354,270]]
S11=[[237,322],[296,297]]
S12=[[176,350],[237,322]]
S13=[[129,370],[176,350]]
S14=[[144,430],[136,411],[125,380],[129,370]]
S2=[[702,118],[787,79]]
S4S3=[[578,173],[608,157],[660,135],[702,118]]
S5=[[507,203],[578,173]]
S6=[[461,223],[507,203]]
S7=[[425,239],[461,223]]
S8=[[382,259],[425,239]]
S9=[[354,270],[382,259]]
T1=[[812,136],[807,120],[795,93]]
T10=[[301,309],[360,284]]
```

```
T11=[[242,335],[253,332],[265,328],[276,322],[301,309]]
T12=[[176,366],[242,335]]
T13=[[143,382],[176,366]]
T14=[[158,427],[150,405],[143,382]]
T21=[[708,131],[721,125],[795,93]]
T22=[[721,125],[795,93]]
T4T3=[[583,184],[600,177],[664,150],[668,148]]
T3Z4=[[668,148],[698,135]]
T42=[[698,135],[708,131]]
T5=[[512,216],[583,184]]
T6=[[466,236],[512,216]]
T7=[[431,252],[466,236]]
T8=[[387,272],[431,252]]
T9=[[360,284],[387,272]]
Y1=[[814,175],[804,157],[799,143]]
Y1N1T1=[[799,143],[812,136]]
Y2=[[756,202],[747,178],[745,167]]
Y4=[[735,214],[725,189],[718,178]]
Y5=[[309,398],[301,377],[294,364],[298,361]]
Y6=[[297,402],[289,382],[283,369],[298,361]]
Y7=[[286,408],[276,388],[271,375],[298,361]]
Z2=[[721,125],[734,152],[745,167]]
Z3=[[708,131],[722,158],[728,173]]
Z4=[[698,135],[710,163],[718,178]]
Z5=[[276,322],[288,348],[294,364],[298,361]]
Z6=[[265,328],[276,354],[283,369],[298,361]]
Z7=[[253,332],[264,360],[271,375],[298,361]]
Z8=[[176,366],[191,399],[196,408],[202,404],[298,361]]

#--------------------COORDENADAS FINGERS Y ACCESO A ELLOS----------------------------

Fingers=[[304,561],[327,549],[351,539],[368,531],[377,521],[375,516],[371,508],[389,479],[396,475],[403,472],[410,469],[418,466],[427,462],[438,456],[446,453],[456,449],[4
62,445],[467,440], [466,428],[459,429],[453,430],[446,435],[437,438],[425,444],[416,448],[406,452],[400,455],[391,458],[383,461], [351,459],[347,450],[343,441],
[331,440],[319,446],[306,451],[293,456],[284,461],[275,465],[263,470],[252,474]]
Sfinger=[[342,571],[337,561],[357,551],[374,544],[412,526],[402,505],[400,501],[398,494],[403,492],[409,489],[418,485],[426,481],[434,478],[441,474],[452,469],[462,464],[47
0,461],[486,455],[500,421],[495,406],[447,401],[438,404],[425,410],[417,413],[406,419],[395,424],[389,426],[381,430],[375,432],[372,428],[369,421],[361,404],[345,411],[312,
426],[299,432],[292,436],[284,439],[267,447],[257,452],[247,455]]

#--------------------PUNTOS PARADA OBLIGADA-----------------------------------------

parada_obligada=[(835,149),(491,293),(807,120),(288,348),(276,354),(264,360),(301,377),(289,382),(276,388),(600,177),(664,150),(608,157),(660,135),(808,199),(820,149)
,(804,157),(191,399),(150,405),(136,411),(173,442),(159,453),(561,309),(596,294),(568,285),(603,270),(789,401),(777,408),(763,418),(321,608),(337,601),(352,600),(485,5
24),(529,503),(511,441),(534,491),(533,382),(551,426),(733,415)]
parada_obligada_uni1=[(533,382),(561,309),(568,285)]
parada_obligada_uni2=[(551,426),(596,294),(603,270)]

#--------------------CLASIFICACION POR TIPOLOGIA DE CARRETERAS----------------------
carreteras_acceso_fingers=[GATEIS,L81,L82,L91,L92,L101,L102,L111,L112,L113,L114,L115,L116,L121,L122,L131,L132,L133,L134,L141,L142,L143,M15M16L14,GATEA
S,GATEBS,GATECS,GATEDS,HS1,GS1,FS1,ES1,Q81,Q82,Q83,Q84,Q85,Q86,Q71,Q72,Q73,LS1,MS1,Q9,Q101,Q102,GATEQS,GATEPS,GATENS,GATEKS1,GATEKS
2,GATEKS3,GATEKS4]
carreteras=
[M2N2,M1a,M125R,S1,S2S1,T2S1,EnlaceP3P4,AS,BS,CN,CS,D2D1,D3,DS,E2E1,E31,E32,E4,E51,E52,EN,ES,FN,FS,GATEKS1,GATEKS2,GATEKS3,GATEKS4,G1,G10,
G11,G2,G3,GN,GS,HN,HS,J10,J5,J6,J7,J8,J9,JN,K10,K11G12,K2,K3,K4,K5,K6,K8,K9,KN,LN,LS,LS1,M1,M10,M11,M12,M13,M14,M15,M16,M2,M31,M32,M3N3_1,M3N3_
2,M5M4,M6,M7,M8,M9,N1,N10,N11,N12,N131,N132,N133,N134,N14,N15,N16,N21,N22,N3N4,N31,N32,N5,N4,N6,N6N5,N7,N8,N9,NM,NS,PISTA1,PISTA2,PISTA3,PN,P
S,QS,RN,S10,S11,S12,S13,S14,S2,S4S3,S5,S6,S7,S8,S9,T1,T10,T11,T12,T13,T14,T21,T22,T42,T4T3,T3Z4,T5,T6,T7,T8,T9,Y1,Y1N1T1,Y2,Y4,Y5,Y6,Y7,Z2,Z3,Z4,Z5,Z6
,Z7,Z8]
carreteras_salida_pista=[G4,G5,G6,G7,G8,G9,R1,R2,R3,R4,R5,R6,P1,P2,P3,P4,P5,P6]

#--------------------FUNCION CARGAR IMAGEN-----------------------------------------

def cargar_imagen(filename, transparent=False):
    image = pygame.image.load(filename)
    image = image.convert()
    if transparent:
        color = image.get_at((0,0))
        image.set_colorkey(color, RLEACCEL)
    return image

#--------------------CLASE CURSOR----------------------------------------------------
```

```
class cursor (pygame.Rect):
	def __init__(self):
		pygame.Rect.__init__(self,0,0,1,1)
	def update(self):
		self.left,self.top=pygame.mouse.get_pos()

#------------------CLASE BOTON----------------------------------------------------

class Boton (pygame.sprite.Sprite):
	def __init__(self, imagen1,imagen2,x=50,y=50):
		super(Boton,self).__init__()
		self.imagen_normal=imagen1
		self.imagen_seleccion=imagen2
		self.imagen_actual=self.imagen_normal
		self.rect=self.imagen_actual.get_rect()
		self.rect.left,self.rect.top=(x,y)
		self.usada="No"
	def update(self,cursor=None):
		if cursor.colliderect(self.rect):
			self.imagen_actual=self.imagen_seleccion
		else:
			self.imagen_actual=self.imagen_normal

#------------------CLASE AVION-----------------------------------

class Avion(pygame.sprite.Sprite):
	def __init__(self):
		super().__init__()
		pygame.sprite.Sprite.__init__(self)
		self.imagen_avion="images/avion.png"
		self.image = cargar_imagen(self.imagen_avion, True)
		self.rect = self.image.get_rect()
		self.rect.centerx = 0
		self.rect.centery = 0
		self.speed=0.009
		self.ruta=[[0,0]]
		self.m = 0
		self.b = 0
		self.carretera=[[0,0]]
		self.destino=[331,631]
		self.zona_aparcamiento="No"
		self.salida_pista="No"
		self.estado="Parado"
		self.ultima_parada=(0,0)
		num_companyia=random.randint(0,len(CompanyiasaereasT1)-1)
		self.nombre_avion=CompanyiasaereasT1[num_companyia]+str(random.randint(1000,9999))
		self.instrucciones=[]
		self.botones_instrucciones=5
		self.auxiliar=[0,0]

	def update(self,time,carreteras_usadas,aviones,TIPO_PROGRAMA):

	#------------------ESTADO AVANZANDO----------------------------

		if self.estado=="Avanzando":
			if self.speed<0.009:
				self.speed=self.speed+0.001
			i=0

			#------------------ Algoritmo no colisionar ----------------------------

			if self.salida_pista=="Si":
				carreteras_disponibles=carreteras_salida_pista
			elif self.zona_aparcamiento=="Si":
				carreteras_disponibles=carreteras+carreteras_acceso_fingers
			else:
				carreteras_disponibles=carreteras
			posibles_carreteras = []
```

```
                for element in carreteras_disponibles:
                        if element[-1]==self.ruta[-1]:
                                posibles_carreteras.append(element)
                        elif element[0]==self.ruta[-1]:
                                posibles_carreteras.append(list(reversed(element)))

                if sqrt((self.ruta[-1][0]-self.rect.centerx)**2+(self.ruta[-1][1]-self.rect.centery)**2)<15 and sqrt((self.ruta[-1][0]-self.rect.centerx)**2+(self.ruta[-1][1]-self.rect.centery)**2)>8:
                        for comparacion in aviones:
                                if comparacion.nombre_avion!=self.nombre_avion and comparacion.carretera in posibles_carreteras and comparacion.estado!="Parado" and sqrt((comparacion.ruta[-1][0]-comparacion.rect.centerx)**2+(comparacion.ruta[-1][1]-comparacion.rect.centery)**2)<25:
                                        self.estado="Parado"


                #------------------------SOLUCION CONFLICTO DESTINO LADO OPUESTO DE T1 --------------------------

                if ([self.rect.centerx,self.rect.centery] in Fingers[:18] or (self.rect.centerx<=502 and self.rect.centery>=441)) and self.destino==[298,361]:
                        self.auxiliar=self.destino
                        self.destino=[462,333]

                elif ([self.rect.centerx,self.rect.centery] in Fingers[18:] or (self.rect.centerx<=511 and self.rect.centery<=423)) and (self.destino==[331,631] or self.destino==[795,426]):
                        self.auxiliar=self.destino
                        self.destino=[1000,1000]

                elif self.destino==[795,426] and self.rect.centerx<473 and self.rect.centery>479:
                        self.auxiliar=self.destino
                        self.destino=[485,524]
                if [self.rect.centerx,self.rect.centery]==[485,524] and self.auxiliar==[795,426]:
                        self.destino=self.auxiliar
                        self.estado="Avanzando"

                if ([self.rect.centerx,self.rect.centery]==[462,333] and self.auxiliar==[298,361]) or (self.rect.centerx>477 and self.rect.centery>393 and self.auxiliar==[331,631] and self.destino==[1000,1000]) or (self.rect.centerx>477 and self.rect.centery>393 and self.auxiliar==[795,426] and self.destino==[1000,1000]):
                        self.destino=self.auxiliar
                        self.estado="Avanzando"



                #-------------------------ALGORITMO AVANCE (RECTAS)----------------------------------------------------
                if self.speed>0.0095:
                        self.speed=self.speed-0.001
                if self.rect.centerx!=self.ruta[-1][0] or self.rect.centery!=self.ruta[-1][1]:
                        for elem in parada_obligada:
                                if list(elem) in self.ruta and elem!=self.ultima_parada and self.ultima_parada not in parada_obligada_uni1+parada_obligada_uni2:
                                        dist_po=sqrt((elem[0]-self.rect.centerx)**2+(elem[1]-self.rect.centery)**2)
                                        if dist_po<9:
                                                self.estado="Parado"
                                                self.ultima_parada=elem
                                                if self.rect.centerx<365 and self.rect.centery>575:
                                                        self.instrucciones.append((str(self.nombre_avion)+" Hold short for runway 07R clearance"))
                                                elif self.rect.centerx>745 and self.rect.centery>395:
                                                        self.instrucciones.append((str(self.nombre_avion)+" Hold short for runway 25L clearance"))
                                                elif self.rect.centerx>780 and self.rect.centery<170:
                                                        self.instrucciones.append((str(self.nombre_avion)+" Hold short for runway 25R clearance"))
                                                elif self.rect.centerx<330 and self.rect.centery<400 and self.rect.centerx>182 and self.rect.centery>350:
                                                        self.instrucciones.append((str(self.nombre_avion)+" Hold short for runway 07L clearance"))
                                                else:
                                                        self.instrucciones.append((str(self.nombre_avion)+" Hold short for clearance"))
                                elif list(elem) in self.ruta and elem!=self.ultima_parada and self.ultima_parada in parada_obligada_uni1:
```

```
                                                    if elem!=parada_obligada_uni2[parada_obligada_uni1.index(self.ultima_parada)]:
                                                        dist_po=sqrt((elem[0]-self.rect.centerx)**2+(elem[1]-self.rect.centery)**2)
                                                        if dist_po<9:
                                                            self.estado="Parado"
                                                            self.instrucciones.append((str(self.nombre_avion)+"      Hold
short for clearance"))
                                                            self.ultima_parada=elem
                                                elif list(elem) in self.ruta and elem!=self.ultima_parada and self.ultima_parada in
parada_obligada_uni2:
                                                    if elem!=parada_obligada_uni1[parada_obligada_uni2.index(self.ultima_parada)]:
                                                        dist_po=sqrt((elem[0]-self.rect.centerx)**2+(elem[1]-self.rect.centery)**2)
                                                        if dist_po<9:
                                                            self.estado="Parado"
                                                            self.instrucciones.append((str(self.nombre_avion)+"      Hold
short for clearance"))
                                                            self.ultima_parada=elem

                                        #--------------------CALCULO RUTA DE DESPEGUE-------------------------

                                        if (self.rect.centerx==self.destino[0] and self.rect.centery==self.destino[1]) and self.destino not in Sfinger:
                                            self.estado="Despegando"
                                            self.imagen_avion="images/avion_pista.png"
                                            #seleccionar ruta segun punto de partida
                                            if self.destino==[331,631]:
                                                self.ruta=[[331,631],[657,487],[795,426]]
                                            elif self.destino==[795,426]:
                                                self.ruta=[[795,426],[483,564],[331,631]]
                                            elif self.destino==[799,143]:
                                                self.ruta=[[799,143],[427,305],[202,404]]
                                            elif self.destino==[298,361]:
                                                self.ruta=[[298,361],[637,214],[799,143]]
                                            self.estado="Parado"
                                        #-------------------------------------------------------------------------------

                                        if self.rect.centerx==self.ruta[i][0] and self.rect.centery==self.ruta[i][1]:
                                            self.ruta=self.ruta[1:]
                                            if self.ruta[i][0]!=self.rect.centerx:
                                                self.m = (self.ruta[i][1]-self.rect.centery)/(self.ruta[i][0]-self.rect.centerx)
                                                self.b = self.ruta[i][1]-self.m*self.ruta[i][0]

                                        if self.ruta[i][0]-self.rect.centerx!=0 and abs(self.ruta[i][0]-self.rect.centerx)>=abs(self.ruta[i][1]-self.rect.centery):
                                            if abs(self.rect.centerx-self.ruta[i][0])<2 and abs(self.rect.centery-self.ruta[i][1])<2:
                                                self.rect.centerx=self.ruta[i][0]
                                                self.rect.centery=self.ruta[i][1]
                                            else:
                                                if self.ruta[i][0]>self.rect.centerx:
                                                    self.rect.centerx+=self.speed*time
                                                    self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,
True),-degrees(atan(self.m))-45)

                                                elif self.ruta[i][0]<=self.rect.centerx:
                                                    self.rect.centerx-=self.speed*time
                                                    self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,
True),-degrees(atan(self.m))-45+180)

                                                self.rect.centery=self.m*self.rect.centerx+self.b

                                        elif self.ruta[i][0]-self.rect.centerx!=0 and abs(self.ruta[i][0]-self.rect.centerx)<abs(self.ruta[i][1]-self.rect.centery):
                                            if abs(self.rect.centerx-self.ruta[i][0])<2 and abs(self.rect.centery-self.ruta[i][1])<2:
                                                self.rect.centerx=self.ruta[i][0]
                                                self.rect.centery=self.ruta[i][1]
                                            else:
                                                if self.ruta[i][1]>self.rect.centery:
                                                    self.rect.centery+=self.speed*time
                                                    if self.ruta[i][0]>self.rect.centerx:

                        self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion, True),-degrees(atan(self.m))-45)
                                                    else:
```

```
            self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion, True),-degrees(atan(self.m))-45+180)

                                                                elif self.ruta[i][1]<=self.rect.centery:
                                                                        self.rect.centery-=self.speed*time
                                                                        if self.ruta[i][0]>self.rect.centerx:

            self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion, True),-degrees(atan(self.m))-45)
                                                                        else:

            self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion, True),-degrees(atan(self.m))-45+180)

                                                                self.rect.centerx=(self.rect.centery-self.b)/self.m

                                                    else:
                                                            if abs(self.rect.centerx-self.ruta[i][0])<2 and abs(self.rect.centery-self.ruta[i][1])<2:
                                                                    self.rect.centerx=self.ruta[i][0]
                                                                    self.rect.centery=self.ruta[i][1]
                                                            else:
                                                                    if self.ruta[i][1]>self.rect.centery:
                                                                            self.rect.centery+=self.speed*time
                                                                            if self.ruta[i][0]>self.rect.centerx:

            self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion, True),-degrees(atan(self.m))-45)
                                                                        else:

            self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion, True),-degrees(atan(self.m))-45+180)
                                                                elif self.ruta[i][1]<=self.rect.centery:
                                                                        self.rect.centery-=self.speed*time
                                                                        self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,
True),-degrees(atan(self.m))-45+180)

                                            #-------------------CALCULO SIGUIENTE RUTA-------------------------------

                                            if sqrt((self.rect.centerx-self.ruta[-1][0])**2+(self.rect.centery-self.ruta[-1][1])**2)<3:

                                                    #-------------------SELECCION POSIBLES CARRETERAS----------------------

                                                    if self.salida_pista=="Si":
                                                            carreteras_disponibles=carreteras_salida_pista
                                                    elif self.zona_aparcamiento=="Si":
                                                            carreteras_disponibles=carreteras+carreteras_acceso_fingers
                                                    else:
                                                            if E4 in carreteras_usadas and self.ruta[-1]==[484,383]:
                                                                    carreteras_disponibles=carreteras+carreteras_acceso_fingers
                                                                    self.zona_aparcamiento="Si"
                                                            else:
                                                                    carreteras_disponibles=carreteras

                                                    if self.ruta[-1] in Fingers:
                                                            self.rect.centerx=Fingers[Sfinger.index(self.destino)][0]
                                                            self.rect.centery=Fingers[Sfinger.index(self.destino)][1]
                                                    if self.destino in Sfinger and list(self.ruta[-1])==self.destino:
                                                            self.ruta=[list(self.ruta[-1]),Fingers[Sfinger.index(self.destino)]]
                                                    else:
                                                            posibles_carreteras = []
                                                            for element in carreteras_disponibles:
                                                                    if element[0]==self.ruta[-1]:
                                                                            posibles_carreteras.append(element)
                                                                    elif element[-1]==self.ruta[-1]:
                                                                            posibles_carreteras.append(list(reversed(element)))

                                                            #-------------------SELECCION POR DISTANCIAS ENTRE LAS CARRETERAS CON PUNTO COMUN
Y VACIAS----------------------------

                                                            if len(posibles_carreteras)>0:
                                                                    posibles_carreteras_ordenada=[posibles_carreteras[0]]
                                                                    dist_x=posibles_carreteras_ordenada[0][-1][0]-self.destino[0]
                                                                    dist_y=posibles_carreteras_ordenada[0][-1][1]-self.destino[1]
                                                                    carreterasAuxiliar=[]
```

```
                                                    for el in carreteras:
                                                        for e in el:
                                                            carreterasAuxiliar.append(e)
                                                    if dist_x==dist_y==0:
                                                        distancias_carreteras=[0]
                                                    elif self.destino not in Sfinger and self.zona_aparcamiento=="Si" and
posibles_carreteras_ordenada[0][-1] in carreterasAuxiliar:
                                                        distancias_carreteras=[-1/(sqrt((dist_x)**2+(dist_y)**2))]
                                                    else:
                                                        distancias_carreteras=[(sqrt((dist_x)**2+(dist_y)**2))]
                                                    posibles_carreteras=posibles_carreteras[1:]
                                                    for e in posibles_carreteras:
                                                        dist_x=e[-1][0]-self.destino[0]
                                                        dist_y=e[-1][1]-self.destino[1]
                                                        if dist_y==0 and dist_x==0:
                                                            dist_e=0
                                                        elif self.destino not in Sfinger and self.zona_aparcamiento=="Si" and e[-1]
in carreterasAuxiliar:
                                                            dist_e=-1/(sqrt((dist_x)**2+(dist_y)**2))
                                                        else:
                                                            dist_e=(sqrt((dist_x)**2+(dist_y)**2))

                                                        j=0
                                                        while j<len(distancias_carreteras):
                                                            if dist_e<distancias_carreteras[j]:

            distancias_carreteras=distancias_carreteras[:j]+[dist_e]+distancias_carreteras[j:]

            posibles_carreteras_ordenada=posibles_carreteras_ordenada[:j]+[e]+posibles_carreteras_ordenada[j:]
                                                                break
                                                            j=j+1
                                                            if j==len(distancias_carreteras):

            distancias_carreteras=distancias_carreteras[:j]+[dist_e]

            posibles_carreteras_ordenada=posibles_carreteras_ordenada[:j]+[e]
                                                                break
                                                    if TIPO_PROGRAMA=="Nivel 4":
                                                        for pos in posibles_carreteras_ordenada:
                                                            if pos not in carreteras_usadas and list(reversed(pos)) not in
carreteras_usadas:
                                                                self.ruta=pos
                                                                self.salida_pista="No"
                                                                self.estado="Avanzando"
                                                                if self.ruta in carreteras:
                                                                    if self.destino not in Sfinger:

            self.zona_aparcamiento="No"
                                                                self.carretera=pos
                                                                break
                                                        if self.ruta not in posibles_carreteras_ordenada and
list(reversed(self.ruta)) not in posibles_carreteras_ordenada:
                                                            self.estado="Parado"
                                                            self.instrucciones.append((str(self.nombre_avion)+" Hold
short for clearance"))

                                                    elif TIPO_PROGRAMA=="Nivel 2":
                                                        pos=posibles_carreteras_ordenada[0]
                                                        if pos not in carreteras_usadas and list(reversed(pos)) not in
carreteras_usadas:
                                                            self.ruta=pos
                                                            self.salida_pista="No"
                                                            self.estado="Avanzando"
                                                            if self.ruta in carreteras:
                                                                if self.destino not in Sfinger:
                                                                    self.zona_aparcamiento="No"
                                                            self.carretera=pos
                                                        if self.ruta not in posibles_carreteras_ordenada and
list(reversed(self.ruta)) not in posibles_carreteras_ordenada:
```

```
                                                                                self.estado="Parado"
                                                                                self.instrucciones.append((str(self.nombre_avion)+"      Hold
short for clearance"))

                                                        else:
                                                                self.estado="Parado"
                                                                self.instrucciones.append((str(self.nombre_avion)+" Hold short for taxi route"))

                        #--------------------ESTADO DESPEGANDO----------------------

                        elif self.estado=="Despegando":
                                i=0
                                self.speed=self.speed+0.00025
                                if sqrt((self.rect.centerx-657)**2+(self.rect.centery-487)**2)<3 and self.ruta[-1]==[795,426]:
                                        self.kill()
                                elif sqrt((self.rect.centerx-483)**2+(self.rect.centery-564)**2)<3 and self.ruta[-1]==[331,631]:
                                        self.kill()
                                elif sqrt((self.rect.centerx-637)**2+(self.rect.centery-214)**2)<3 and self.ruta[-1]==[799,143]:
                                        self.kill()
                                elif sqrt((self.rect.centerx-427)**2+(self.rect.centery-305)**2)<3 and self.ruta[-1]==[202,404]:
                                        self.kill()
                                if self.rect.centerx!=self.ruta[-1][0] or self.rect.centery!=self.ruta[-1][1]:

                                        if self.rect.centerx==self.ruta[i][0] and self.rect.centery==self.ruta[i][1]:
                                                self.ruta=self.ruta[1:]
                                                self.m = (self.ruta[i][1]-self.rect.centery)/(self.ruta[i][0]-self.rect.centerx)
                                                self.b = self.ruta[i][1]-self.m*self.ruta[i][0]

                                        if self.ruta[i][0]-self.rect.centerx!=0 and abs(self.ruta[i][0]-self.rect.centerx)>=abs(self.ruta[i][1]-self.rect.centery):

                                                if self.ruta[i][0]>self.rect.centerx:
                                                        self.rect.centerx+=self.speed*time
                                                        self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,          True),-
degrees(atan(self.m))-45)

                                                else:
                                                        self.rect.centerx-=self.speed*time
                                                        self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,          True),-
degrees(atan(self.m))-45+180)
                                                self.rect.centery=self.m*self.rect.centerx+self.b

                                        elif self.ruta[i][0]-self.rect.centerx!=0 and abs(self.ruta[i][0]-self.rect.centerx)<abs(self.ruta[i][1]-self.rect.centery):
                                                if self.ruta[i][1]>self.rect.centery:
                                                        self.rect.centery+=self.speed*time
                                                        self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,          True),-
degrees(atan(self.m))-45)
                                                else:
                                                        self.rect.centery-=self.speed*time
                                                        self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,          True),-
degrees(atan(self.m))-45+180)
                                                self.rect.centerx=(self.rect.centery-self.b)/self.m

                                        else:

                                                if self.ruta[i][1]>self.rect.centery:
                                                        self.rect.centery+=self.speed*time
                                                        self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,          True),-
degrees(atan(self.m))-45)
                                                else:
                                                        self.rect.centery-=self.speed*time
                                                        self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,          True),-
degrees(atan(self.m))-45+180)

                        #--------------------ESTADO ATERRIZANDO---------------------

                        elif self.estado=="Aterrizando":
                                i=0
                                if list(self.rect.center)==self.ruta[-1]:
                                        self.estado="Avanzando"
```

```
                                        if self.speed>0.015:
                                                self.speed=self.speed-0.00020
                                        if abs(self.rect.centerx-self.ruta[i][0])<4 and abs(self.rect.centery-self.ruta[i][1])<4:
                                                        self.rect.centerx=self.ruta[i][0]
                                                        self.rect.centery=self.ruta[i][1]
                                        if self.rect.centerx!=self.ruta[-1][0] or self.rect.centery!=self.ruta[-1][1]:

                                                if self.rect.centerx==self.ruta[i][0] and self.rect.centery==self.ruta[i][1]:
                                                        self.ruta=self.ruta[1:]
                                                        self.m = (self.ruta[i][1]-self.rect.centery)/(self.ruta[i][0]-self.rect.centerx)
                                                        self.b = self.ruta[i][1]-self.m*self.ruta[i][0]

                                                if self.ruta[i][0]-self.rect.centerx!=0 and abs(self.ruta[i][0]-self.rect.centerx)>=abs(self.ruta[i][1]-self.rect.centery):

                                                        if self.ruta[i][0]>self.rect.centerx:
                                                                self.rect.centerx+=self.speed*time
                                                                self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,  True),-
degrees(atan(self.m))-45)

                                                        else:
                                                                self.rect.centerx-=self.speed*time
                                                                self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,  True),-
degrees(atan(self.m))-45+180)
                                                        self.rect.centery=self.m*self.rect.centerx+self.b

                                                elif self.ruta[i][0]-self.rect.centerx!=0 and abs(self.ruta[i][0]-self.rect.centerx)<abs(self.ruta[i][1]-self.rect.centery):
                                                        if self.ruta[i][1]>self.rect.centery:
                                                                self.rect.centery+=self.speed*time
                                                                self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,  True),-
degrees(atan(self.m))-45)
                                                        else:
                                                                self.rect.centery-=self.speed*time
                                                                self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,  True),-
degrees(atan(self.m))-45+180)
                                                        self.rect.centerx=(self.rect.centery-self.b)/self.m

                                                else:

                                                        if self.ruta[i][1]>self.rect.centery:
                                                                self.rect.centery+=self.speed*time
                                                                self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,  True),-
degrees(atan(self.m))-45)
                                                        else:
                                                                self.rect.centery-=self.speed*time
                                                                self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion,  True),-
degrees(atan(self.m))-45+180)

                                #--------------------ESTADO PARADO-------------------------

                                elif self.estado=="Parado":
                                        self.speed=0
                                        if list(self.rect.center) in Fingers:
                                                self.ruta=[list(self.rect.center),Sfinger[Fingers.index(list(self.rect.center))]]
                                                self.carretera=[[0,0]]

                                        i=0
                                        if self.ruta[i][0]>self.rect.centerx:
                                                self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion, True),-degrees(atan(self.m))-45)
                                        else:
                                                self.image=pygame.transform.rotate(cargar_imagen(self.imagen_avion, True),-degrees(atan(self.m))-45+180)

                                        pass


#--------------------PROGRAMA PRINCIPAL----------------------------

def main():
```

```
#--------------------CARGA DE IMAGENES E INICIACION VARIABLES--------------------

TIPO_PROGRAMA="Nivel 4"
fuente=pygame.font.Font(None,22)
fuente2=pygame.font.Font(None,18)
clock = pygame.time.Clock()
pantalla=pygame.display.set_mode((1275,650))
fondo=cargar_imagen('images/T1.png')
fondo_negro=cargar_imagen('images/fondo_negro.png')
aviones=pygame.sprite.Group()
imagen_finger_no_usado=cargar_imagen('images/finger.png',True)
imagen_finger_usado=cargar_imagen('images/finger_usado.png')
fingers_seleccion=pygame.sprite.Group()
for element in Fingers:
        fingers_seleccion.add(Boton(imagen_finger_no_usado,imagen_finger_usado,element[0],element[1]))
avionseleccionado=pygame.sprite.GroupSingle()
boton_cross=cargar_imagen('images/botones.png')
boton_cross_sel=cargar_imagen('images/botones_sel.png')
boton_inhabilitado=cargar_imagen('images/botones_inhabilitado.png')
imagen_pista1=cargar_imagen('images/07L-25R.png',True)
imagen_pista1_sel=cargar_imagen('images/07L-25R_sel.png',True)
imagen_pista3=cargar_imagen('images/07R-25L.png',True)
imagen_pista3_sel=cargar_imagen('images/07R-25L_sel.png',True)
boton_07L=cargar_imagen('images/07L.png')
boton_07L_sel=cargar_imagen('images/07L_sel.png')
boton_07R=cargar_imagen('images/07R.png')
boton_07R_sel=cargar_imagen('images/07R_sel.png')
boton_02=cargar_imagen('images/02.png')
boton_02_sel=cargar_imagen('images/02_sel.png')
boton_20=cargar_imagen('images/20.png')
boton_20_sel=cargar_imagen('images/20_sel.png')
boton_25R=cargar_imagen('images/25R.png')
boton_25R_sel=cargar_imagen('images/25R_sel.png')
boton_25L=cargar_imagen('images/25L.png')
boton_25L_sel=cargar_imagen('images/25L_sel.png')
boton_up=cargar_imagen('images/up.png')
boton_up_sel=cargar_imagen('images/up_marcada.png')
boton_down=cargar_imagen('images/down.png')
boton_down_sel=cargar_imagen('images/down_marcada.png')
cursor1=cursor()
botoncross=Boton(boton_cross,boton_cross_sel,992,85)
botonHS=Boton(boton_cross,boton_cross_sel,992,125)
botonLUAW=Boton(boton_cross,boton_cross_sel,992,165)
botonCFT=Boton(boton_cross,boton_cross_sel,992,205)
boton07L=Boton(boton_07L,boton_07L_sel,94,421)
boton07R=Boton(boton_07R,boton_07R_sel,274,594)
boton02=Boton(boton_02,boton_02_sel,531,551)
boton20=Boton(boton_20,boton_20_sel,677,43)
boton25L=Boton(boton_25L,boton_25L_sel,807,395)
boton25R=Boton(boton_25R,boton_25R_sel,834,101)
botonup=Boton(boton_up,boton_up_sel,1239,605)
botondown=Boton(boton_down,boton_down_sel,1239,620)
boton_pista1=Boton(imagen_pista1,imagen_pista1_sel,0,0)
boton_pista3=Boton(imagen_pista3,imagen_pista3_sel,0,0)
carreteras_usadas=[]
destino_avion=fuente.render("No seleccionat",0,(255,255,255))
Permiso_aparcamiento=fuente.render("No seleccionat",0,(255,255,255))
Nombre_avion=fuente.render("",0,(255,255,255))
Orden_ATC1=fuente2.render("",0,(255,255,255))
Orden_ATC2=fuente2.render("",0,(255,255,255))
Orden_ATC3=fuente2.render("",0,(255,255,255))
Orden_ATC4=fuente2.render("",0,(255,255,255))
linea_conversacion_ATC=-1
fingersocupadosa=[]
fingersocupadosb=[]
while True:

        time = clock.tick(7)
        keys=pygame.key.get_pressed()
```

```
#--------------------ACTUALIZACION IMAGEN PISTAS--------------------

usando_pista1=0
usando_pista3=0
for element in aviones:
	if element.rect.centery<415 and (element.estado=="Despegando" or element.estado=="Aterrizando"):
		usando_pista1=usando_pista1+1
	elif element.rect.centery>415 and (element.estado=="Despegando" or element.estado=="Aterrizando"):
		usando_pista3=usando_pista3+1

for eventos in pygame.event.get():
	if eventos.type==QUIT:
		sys.exit()

	#--------------------CAMBIO NIVEL A-SMGCS-------------------------

	if TIPO_PROGRAMA=="Nivel 2" and keys[K_n]:
		TIPO_PROGRAMA="Nivel 4"
	elif TIPO_PROGRAMA=="Nivel 4" and keys[K_n]:
		TIPO_PROGRAMA="Nivel 2"

	#--------------------DESELECCIONAR AVIO--------------------
	if pygame.mouse.get_pressed()[2]==True and len(avionseleccionado)>0:
		avionseleccionado.remove(avionseleccionado.sprite)

	#--------------------ACCIONES CON BOTON IZQUIERDO DEL RATON--------------------

	if pygame.mouse.get_pressed()[0]==True:
		avio=Avion()
		if len(avionseleccionado)>0:
			if cursor1.colliderect(botonup.rect) and
abs(linea_conversacion_ATC)+3<len(avionseleccionado.sprite.instrucciones):
				linea_conversacion_ATC=linea_conversacion_ATC-1
			elif cursor1.colliderect(botondown.rect) and linea_conversacion_ATC<-1:
				linea_conversacion_ATC=linea_conversacion_ATC+1
		for element in fingers_seleccion:
			if cursor1.colliderect(element):
				if len(avionseleccionado)>0:

		avionseleccionado.sprite.destino=Sfinger[Fingers.index([element.rect.left,element.rect.top])]
					avionseleccionado.sprite.zona_aparcamiento="Si"
				else:
					element.image = cargar_imagen("images/finger_usado.png", True)
					avio.rect.centerx = element.rect.left
					avio.rect.centery = element.rect.top

		avio.ruta=[[avio.rect.centerx,avio.rect.centery],Sfinger[Fingers.index([avio.rect.centerx,avio.rect.centery])]]
					avio.zona_aparcamiento="Si"
					avio.instrucciones.append((str(avio.nombre_avion)+" Hold short for taxi
route"))
					aviones.add(avio)
					avionseleccionado.add(avio)
		if cursor1.colliderect(botoncross.rect) and len(avionseleccionado)>0 and "Line up and wait" not in
avionseleccionado.sprite.instrucciones[-1] and avionseleccionado.sprite.estado!="Despegando" and avionseleccionado.sprite.estado!="Aterrizando" and
avionseleccionado.sprite.estado=="Parado":

			if avionseleccionado.sprite.destino==avionseleccionado.sprite.ruta[-1]:
				if [avionseleccionado.sprite.rect.centerx,avionseleccionado.sprite.rect.centery] not in
Fingers:
					avionseleccionado.sprite.estado="Avanzando"
					avionseleccionado.sprite.botones_instrucciones=1
					if tuple(avionseleccionado.sprite.ruta[0]) in
parada_obligada_uni1+parada_obligada_uni2:

		avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Cross Runway 02/20"))
					elif tuple(avionseleccionado.sprite.ruta[0]) in parada_obligada:
```

```
                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Cross"))

                                                        else:
                                                                avionseleccionado.sprite.estado="Avanzando"
                                                                avionseleccionado.sprite.botones_instrucciones=5
                                                                if tuple(avionseleccionado.sprite.ruta[0]) in
parada_obligada_uni1+parada_obligada_uni2:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Cross Runway 02/20"))
                                                                elif tuple(avionseleccionado.sprite.ruta[0]) in parada_obligada:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Cross"))
                                                                else:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Clear route to advance"))
                                                        elif cursor1.colliderect(botonHS.rect) and len(avionseleccionado)>0 and avionseleccionado.sprite.estado!="Parado"
and avionseleccionado.sprite.estado!="Despegando" and avionseleccionado.sprite.estado!="Aterrizando" and not "Line up and wait" in
avionseleccionado.sprite.instrucciones[-1] and not avionseleccionado.sprite.ruta[0] in G10+G11+K11G12+G1+G2+G3+Y1+Y5+Y6+Y7+N1:
                                                        avionseleccionado.sprite.botones_instrucciones=2
                                                        avionseleccionado.sprite.estado="Parado"
                                                        avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Hold
short for clearance"))
                                                        elif cursor1.colliderect(botonLUAW.rect) and len(avionseleccionado)>0 and
avionseleccionado.sprite.estado=="Parado" and avionseleccionado.sprite.ruta[0] in G10+G11+K11G12+G1+G2+G3+Y1+Y5+Y6+Y7+N1 and ("clearance" in
avionseleccionado.sprite.instrucciones[-1] and "runway" in avionseleccionado.sprite.instrucciones[-1]):
                                                        avionseleccionado.sprite.botones_instrucciones=3
                                                        avionseleccionado.sprite.estado="Avanzando"
                                                        if avionseleccionado.sprite.ruta[0] in G10:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 07R, taxiway G10,Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in G11:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 07R, taxiway G11,Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in K11G12:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 07R, taxiway G12,Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in G1:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 25L, taxiway G1, Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in G2:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 25L, taxiway G2, Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in G3:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 25L, taxiway G1, Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in Y5:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 07L, taxiway Y5, Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in Y6:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 07L, taxiway Y6, Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in Y7:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 07L, taxiway Y7, Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in Y1:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 25R, taxiway Y1, Line up and wait"))
                                                        elif avionseleccionado.sprite.ruta[0] in N1:

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" Runway 25R, taxiway N1, Line up and wait"))
                                                        elif cursor1.colliderect(botonCFT.rect) and len(avionseleccionado)>0 and
avionseleccionado.sprite.estado=="Parado" and ([657,487] in avionseleccionado.sprite.ruta or [483,564] in avionseleccionado.sprite.ruta or [427,305] in
avionseleccionado.sprite.ruta or [637,214] in avionseleccionado.sprite.ruta):
                                                        avionseleccionado.sprite.botones_instrucciones=4
                                                        if usando_pista1==0 and ("07L" in avionseleccionado.sprite.instrucciones[-1] or "25R" in
avionseleccionado.sprite.instrucciones[-1]):
                                                                avionseleccionado.sprite.estado="Despegando"
```

```
                avionseleccionado.sprite.instrucciones.append(str(avionseleccionado.sprite.nombre_avion)+"  Cleared  for  take  off  Runway
"+str(avionseleccionado.sprite.instrucciones[-1][-33:-30]))
                                                elif  usando_pista3==0  and  ("07R"  in  avionseleccionado.sprite.instrucciones[-1]  or  "25L"  in
avionseleccionado.sprite.instrucciones[-1]):
                                                        avionseleccionado.sprite.estado="Despegando"

                avionseleccionado.sprite.instrucciones.append(str(avionseleccionado.sprite.nombre_avion)+"  Cleared  for  take  off  Runway
"+str(avionseleccionado.sprite.instrucciones[-1][-33:-30]))
                                        elif cursor1.colliderect(boton07L.rect):
                                                if len(avionseleccionado)>0:
                                                        avionseleccionado.sprite.destino=[298,361]

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" new destination Runway 07L"))
                                                else:
                                                        if usando_pista1==0 and len(fingersocupadosa)<22:
                                                                avio=Avion()
                                                                avio.rect.centerx=202
                                                                avio.rect.centery=404
                                                                probabilidad=random.randint(0,10)
                                                                numazar=random.randint(18,len(Sfinger)-1)
                                                                fingerpreparado=0
                                                                while fingerpreparado==0:
                                                                        if Sfinger[numazar] not in fingersocupadosa:
                                                                                avio.destino=Sfinger[numazar]
                                                                                fingerpreparado=1
                                                                                avio.estado="Aterrizando"
                                                                        else:
                                                                                numazar=random.randint(18,len(Sfinger)-1)
                                                                avio.instrucciones.append((str(avio.nombre_avion)+"  Runway  07L
Cleared to land"))
                                                                if probabilidad<6:
                                                                        punto_salida=[463,289]
                                                                        avio.instrucciones.append((str(avio.nombre_avion)+"  Turn
right on Taxiway R4"))
                                                                elif probabilidad==6 or probabilidad==7:
                                                                        punto_salida=[512,267]
                                                                        avio.instrucciones.append((str(avio.nombre_avion)+"  Turn
right on Taxiway R2"))
                                                                elif probabilidad==8:
                                                                        punto_salida=[546,253]
                                                                        avio.instrucciones.append((str(avio.nombre_avion)+"  Turn
left on Taxiway P2"))
                                                                elif probabilidad==9:
                                                                        punto_salida=[602,228]
                                                                        avio.instrucciones.append((str(avio.nombre_avion)+"  Turn
right on Taxiway R1"))
                                                                else:
                                                                        punto_salida=[664,201]
                                                                        avio.instrucciones.append((str(avio.nombre_avion)+"  Turn
left on Taxiway P1"))
                                                                avio.ruta=[[202,404],punto_salida]
                                                                avio.estado="Aterrizando"
                                                                avio.salida_pista="Si"
                                                                avio.speed=0.04
                                                                avio.zona_aparcamiento="Si"
                                                                aviones.add(avio)
                                                                avionseleccionado.add(avio)
                                        elif cursor1.colliderect(boton07R.rect):
                                                if len(avionseleccionado)>0:
                                                        avionseleccionado.sprite.destino=[331,631]

                avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" new destination Runway 07R"))
                                                else:
                                                        if usando_pista3==0 and len(fingersocupadosb)<18:
                                                                avio=Avion()
                                                                avio.rect.centerx=331
                                                                avio.rect.centery=631
                                                                probabilidad=random.randint(0,100)
                                                                numazar=random.randint(0,17)
```

```
                                                                fingerpreparado=0
                                                                while fingerpreparado==0:
                                                                        if Sfinger[numazar] not in fingersocupadosb:
                                                                                avio.destino=Sfinger[numazar]
                                                                                fingerpreparado=1
                                                                                avio.estado="Aterrizando"
                                                                        else:
                                                                                numazar=random.randint(0,17)
                                                                avio.instrucciones.append((str(avio.nombre_avion)+"      Runway      07R
Cleared to land"))
                                                                if probabilidad<90:
                                                                        punto_salida=[603,511]
                                                                        avio.instrucciones.append((str(avio.nombre_avion)+"    Turn
left on Taxiway G6"))
                                                                elif probabilidad>=90 and probabilidad<98:
                                                                        punto_salida=[657,487]
                                                                        avio.instrucciones.append((str(avio.nombre_avion)+"    Turn
left on Taxiway G5"))
                                                                else:
                                                                        punto_salida=[714,462]
                                                                        avio.instrucciones.append((str(avio.nombre_avion)+"    Turn
left on Taxiway G4"))
                                                                avio.ruta=[[331,631],punto_salida]
                                                                avio.estado="Aterrizando"
                                                                avio.salida_pista="Si"
                                                                avio.speed=0.04
                                                                avio.zona_aparcamiento="Si"
                                                                aviones.add(avio)
                                                                avionseleccionado.add(avio)
                                                elif cursor1.colliderect(boton25L.rect):
                                                        if len(avionseleccionado)>0:
                                                                avionseleccionado.sprite.destino=[795,426]

        avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" new destination Runway 25L"))
                                                        else:
                                                                if usando_pista3==0 and len(fingersocupadosb)<18:
                                                                        avio=Avion()
                                                                        avio.rect.centerx=795
                                                                        avio.rect.centery=426
                                                                        probabilidad=random.randint(0,10)
                                                                        numazar=random.randint(0,17)
                                                                        fingerpreparado=0
                                                                        while fingerpreparado==0:
                                                                                if Sfinger[numazar] not in fingersocupadosb:
                                                                                        avio.destino=Sfinger[numazar]
                                                                                        fingerpreparado=1
                                                                                        avio.estado="Aterrizando"
                                                                                else:
                                                                                        numazar=random.randint(0,17)
                                                                        avio.instrucciones.append((str(avio.nombre_avion)+"      Runway      25L
Cleared to land"))
                                                                        if probabilidad<8:
                                                                                punto_salida=[537,540]
                                                                                avio.instrucciones.append((str(avio.nombre_avion)+"    Turn
right on Taxiway G7"))
                                                                        elif probabilidad==8 or probabilidad==9:
                                                                                punto_salida=[483,564]
                                                                                avio.instrucciones.append((str(avio.nombre_avion)+"    Turn
right on Taxiway G8"))
                                                                        else:
                                                                                punto_salida=[419,591]
                                                                                avio.instrucciones.append((str(avio.nombre_avion)+"    Turn
right on Taxiway G9"))
                                                                        avio.ruta=[[795,426],punto_salida]
                                                                        avio.estado="Aterrizando"
                                                                        avio.speed=0.04
                                                                        avio.salida_pista="Si"
                                                                        aviones.add(avio)
                                                                        avio.zona_aparcamiento="Si"
```

```
                                                                avionseleccionado.add(avio)

                                                    elif cursor1.colliderect(boton25R.rect):
                                                            if len(avionseleccionado)>0:
                                                                    avionseleccionado.sprite.destino=[799,143]

        avionseleccionado.sprite.instrucciones.append((str(avionseleccionado.sprite.nombre_avion)+" new destination Runway 25R"))
                                                            else:
                                                                    if usando_pista1==0 and len(fingersocupadosa)<22:
                                                                            avio=Avion()
                                                                            avio.rect.centerx=799
                                                                            avio.rect.centery=143
                                                                            probabilidad=random.randint(0,10)
                                                                            numazar=random.randint(18,len(Sfinger)-1)
                                                                            fingerpreparado=0
                                                                            while fingerpreparado==0:
                                                                                    if Sfinger[numazar] not in fingersocupadosa:
                                                                                            avio.destino=Sfinger[numazar]
                                                                                            fingerpreparado=1
                                                                                            avio.estado="Aterrizando"
                                                                                    else:
                                                                                            numazar=random.randint(18,len(Sfinger)-1)
                                                                            avio.instrucciones.append((str(avio.nombre_avion)+"  Runway  25R
Cleared to land"))
                                                                            if probabilidad<8:
                                                                                    punto_salida=[546,253]
                                                                                    avio.instrucciones.append((str(avio.nombre_avion)+"  Turn
left on Taxiway R3"))
                                                                            elif probabilidad==8 and probabilidad==9:
                                                                                    punto_salida=[486,280]
                                                                                    avio.instrucciones.append((str(avio.nombre_avion)+"  Turn
left on Taxiway R5"))
                                                                            else:
                                                                                    punto_salida=[416,309]
                                                                                    avio.instrucciones.append((str(avio.nombre_avion)+"  Turn
left on Taxiway R6"))
                                                                            avio.ruta=[[799,143],punto_salida]
                                                                            avio.salida_pista="Si"
                                                                            avio.speed=0.04
                                                                            aviones.add(avio)
                                                                            avio.zona_aparcamiento="Si"
                                                                            avionseleccionado.add(avio)
                                    for element in aviones:
                                            if cursor1.colliderect(element):
                                                    avionseleccionado.add(element)
                                                    linea_conversacion_ATC=-1
                                            if element.estado=="Parado":
                                                    element.imagen_avion="images/avion_parado.png"
                                            elif element.estado=="Aterrizando" or element.estado=="Despegando":
                                                    element.imagen_avion="images/avion_pista.png"
                                            else:
                                                    element.imagen_avion="images/avion.png"
                                    if len(avionseleccionado)>0:
                                            avionseleccionado.sprite.imagen_avion="images/avion_seleccionado.png"

                                    carreteras_usadas=[]
                                    fingersocupadosa=[]
                                    fingersocupadosb=[]
                                    usando_pista1=0
                                    usando_pista3=0
                                    for element in aviones:
                                            if element.rect.centery<415 and (element.estado=="Despegando" or element.estado=="Aterrizando"):
                                                    boton_pista1.imagen_actual=boton_pista1.imagen_seleccion
                                                    usando_pista1=usando_pista1+1
                                            elif element.rect.centery>415 and (element.estado=="Despegando" or element.estado=="Aterrizando"):
                                                    boton_pista3.imagen_actual=boton_pista3.imagen_seleccion
                                                    usando_pista3=usando_pista3+1
                                            if element.carretera not in carreteras_usadas:
                                                    carreteras_usadas.append(element.carretera)
```

```
                    if element.destino in Sfinger[18:40]:
                        fingersocupadosa.append(element.destino)
                    elif element.destino in Sfinger[0:18]:
                        fingersocupadosb.append(element.destino)
                    if [element.rect.centerx,element.rect.centery] in Fingers[0:18]:
                        fingersocupadosb.append(Sfinger[Fingers.index([element.rect.centerx,element.rect.centery])])
                    elif [element.rect.centerx,element.rect.centery] in Fingers[18:40]:
                        fingersocupadosa.append(Sfinger[Fingers.index([element.rect.centerx,element.rect.centery])])
                if usando_pista1==0:
                    boton_pista1.imagen_actual=boton_pista1.imagen_normal
                if usando_pista3==0:
                    boton_pista3.imagen_actual=boton_pista3.imagen_normal
                aviones.update(time,carreteras_usadas,aviones,TIPO_PROGRAMA)
                pantalla.blit(fondo_negro,(0,0))
                pantalla.blit(fondo,(0,0))
                for el in fingers_seleccion:
                    el.update(cursor1)
                    pantalla.blit(el.imagen_actual,(el.rect.left,el.rect.top))
                cursor1.update()


                #---------------------------ACTUALIZACION ASPECTO BOTONES---------------------------------------

                if len(avionseleccionado)>0:
                    if avionseleccionado.sprite.estado=="Aterrizando":
                        botoncross.imagen_actual=boton_inhabilitado
                        botonHS.imagen_actual=boton_inhabilitado
                        botonLUAW.imagen_actual=boton_inhabilitado
                        botonCFT.imagen_actual=boton_inhabilitado
                    elif avionseleccionado.sprite.botones_instrucciones==5:
                        if avionseleccionado.sprite.estado=="Parado" and avionseleccionado.sprite.destino in Sfinger :
                            if [avionseleccionado.sprite.rect.centerx,avionseleccionado.sprite.rect.centery] in Fingers:
                                botoncross.imagen_actual=boton_inhabilitado
                                botonHS.imagen_actual=boton_inhabilitado
                                botonLUAW.imagen_actual=boton_inhabilitado
                                botonCFT.imagen_actual=boton_inhabilitado
                            else:
                                botoncross.update(cursor1)
                                botonHS.imagen_actual=boton_inhabilitado
                                botonLUAW.imagen_actual=boton_inhabilitado
                                botonCFT.imagen_actual=boton_inhabilitado
                        elif avionseleccionado.sprite.estado=="Parado" and avionseleccionado.sprite.destino not in Sfinger and
avionseleccionado.sprite.ruta[0] in G10+G11+K11G12+G1+G2+G3+Y1+Y5+Y6+Y7+N1 and ("clearance" in avionseleccionado.sprite.instrucciones[-1] and "runway" in
avionseleccionado.sprite.instrucciones[-1]):
                            botoncross.imagen_actual=boton_inhabilitado
                            botonHS.imagen_actual=boton_inhabilitado
                            botonLUAW.update(cursor1)
                            botonCFT.imagen_actual=boton_inhabilitado
                        elif avionseleccionado.sprite.estado=="Parado":
                            botoncross.update(cursor1)
                            botonHS.imagen_actual=boton_inhabilitado
                            botonLUAW.imagen_actual=boton_inhabilitado
                            botonCFT.imagen_actual=boton_inhabilitado
                        elif avionseleccionado.sprite.ruta[0] in G10+G11+K11G12+G1+G2+G3+Y1+Y5+Y6+Y7+N1:
                            botoncross.imagen_actual=boton_inhabilitado
                            botonHS.imagen_actual=boton_inhabilitado
                            botonLUAW.imagen_actual=boton_inhabilitado
                            botonCFT.imagen_actual=boton_inhabilitado
                        else:
                            botonHS.update(cursor1)
                            botoncross.imagen_actual=boton_inhabilitado
                            botonLUAW.imagen_actual=boton_inhabilitado
                            botonCFT.imagen_actual=boton_inhabilitado
                    elif avionseleccionado.sprite.botones_instrucciones==2:
                        botoncross.update(cursor1)
                        botonHS.imagen_actual=boton_inhabilitado
                        botonLUAW.imagen_actual=boton_inhabilitado
                        botonCFT.imagen_actual=boton_inhabilitado
                    elif avionseleccionado.sprite.botones_instrucciones==1:
```

```
                                        if avionseleccionado.sprite.estado=="Parado" and avionseleccionado.sprite.destino in Sfinger:
                                                botoncross.imagen_actual=boton_inhabilitado
                                                botonHS.imagen_actual=boton_inhabilitado
                                                botonLUAW.imagen_actual=boton_inhabilitado
                                                botonCFT.imagen_actual=boton_inhabilitado
                                        else:
                                                botoncross.imagen_actual=boton_inhabilitado
                                                botonHS.update(cursor1)
                                                botonLUAW.imagen_actual=boton_inhabilitado
                                                botonCFT.imagen_actual=boton_inhabilitado
                                elif avionseleccionado.sprite.botones_instrucciones==3:
                                        if usando_pista1==0 and ("07L" in avionseleccionado.sprite.instrucciones[-1] or "25R" in
avionseleccionado.sprite.instrucciones[-1]):
                                                botonCFT.update(cursor1)
                                        elif usando_pista3==0 and ("07R" in avionseleccionado.sprite.instrucciones[-1] or "25L" in
avionseleccionado.sprite.instrucciones[-1]):
                                                botonCFT.update(cursor1)
                                        else:
                                                botonCFT.imagen_actual=boton_inhabilitado
                                        botoncross.imagen_actual=boton_inhabilitado
                                        botonHS.imagen_actual=boton_inhabilitado
                                        botonLUAW.imagen_actual=boton_inhabilitado

                                elif avionseleccionado.sprite.botones_instrucciones==4:
                                        if avionseleccionado.sprite.estado=="Despegando":
                                                botoncross.imagen_actual=boton_inhabilitado
                                                botonHS.imagen_actual=boton_inhabilitado
                                                botonLUAW.imagen_actual=boton_inhabilitado
                                                botonCFT.imagen_actual=boton_inhabilitado
                                        else:
                                                botoncross.imagen_actual=boton_inhabilitado
                                                botonHS.imagen_actual=boton_inhabilitado
                                                botonLUAW.imagen_actual=boton_inhabilitado
                                                botonCFT.update(cursor1)
                        else:
                                botoncross.update(cursor1)
                                botonHS.update(cursor1)
                                botonLUAW.update(cursor1)
                                botonCFT.update(cursor1)

                        #------------------------------ACTUALIZACION RESTO ELEMENTOS---------------------------------

                        boton07L.update(cursor1)
                        boton07R.update(cursor1)
                        boton02.update(cursor1)
                        boton20.update(cursor1)
                        boton25L.update(cursor1)
                        boton25R.update(cursor1)
                        botonup.update(cursor1)
                        botondown.update(cursor1)
                        pantalla.blit(boton_pista1.imagen_actual,(0,0))
                        pantalla.blit(boton_pista3.imagen_actual,(0,0))
                        pantalla.blit(botoncross.imagen_actual,(992,85))
                        pantalla.blit(botonHS.imagen_actual,(992,125))
                        pantalla.blit(botonLUAW.imagen_actual,(992,165))
                        pantalla.blit(botonCFT.imagen_actual,(992,205))
                        pantalla.blit(boton07L.imagen_actual,(94,421))
                        pantalla.blit(boton07R.imagen_actual,(274,594))
                        pantalla.blit(boton02.imagen_actual,(531,551))
                        pantalla.blit(boton20.imagen_actual,(677,43))
                        pantalla.blit(boton25R.imagen_actual,(834,101))
                        pantalla.blit(boton25L.imagen_actual,(807,395))
                        pantalla.blit(botonup.imagen_actual,(1239,605))
                        pantalla.blit(botondown.imagen_actual,(1239,620))
                        aviones.draw(pantalla)

                        #-------------------ESCRIBIR TEXTOS--------------------

                        if len(avionseleccionado)>0:
```

```
                                if avionseleccionado.sprite.destino==[331,631] or avionseleccionado.sprite.auxiliar==[331,631]:
                                        destino_avion=fuente.render("07R",0,(255,255,255))
                                elif avionseleccionado.sprite.destino==[795,426] or avionseleccionado.sprite.auxiliar==[795,426]:
                                        destino_avion=fuente.render("25L",0,(255,255,255))
                                elif avionseleccionado.sprite.destino==[298,361] or avionseleccionado.sprite.auxiliar==[298,361]:
                                        destino_avion=fuente.render("07L",0,(255,255,255))
                                elif avionseleccionado.sprite.destino==[799,143] or avionseleccionado.sprite.auxiliar==[799,143]:
                                        destino_avion=fuente.render("25R",0,(255,255,255))
                                elif avionseleccionado.sprite.destino in Sfinger:
                                        destino_avion=fuente.render("F"+str(Sfinger.index(avionseleccionado.sprite.destino)+1),0,(255,255,255))
                                Permiso_aparcamiento=fuente.render(str(avionseleccionado.sprite.salida_pista),0,(255,255,255))
                                if avionseleccionado.sprite.estado=="Parado":
                                        Velocidad_avion=fuente.render(str(int(0)),0,(255,255,255))
                                else:
                                        Velocidad_avion=fuente.render(str(int(avionseleccionado.sprite.speed*3333.33333)),0,(255,255,255))
                                Estado_Avion=fuente.render(str(avionseleccionado.sprite.estado),0,(255,255,255))
                                Nombre_avion=fuente.render(avionseleccionado.sprite.nombre_avion,0,(255,255,255))
                                if len(avionseleccionado.sprite.instrucciones)==1:
                                        Orden_ATC1=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC]),0,(255,255,255))
                                        Orden_ATC2=fuente2.render("",0,(255,255,255))
                                        Orden_ATC3=fuente2.render("",0,(255,255,255))
                                        Orden_ATC4=fuente2.render("",0,(255,255,255))
                                elif len(avionseleccionado.sprite.instrucciones)==2:
                                        Orden_ATC1=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC]),0,(255,255,255))
                                        Orden_ATC2=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC-
1]),0,(255,255,255))
                                        Orden_ATC3=fuente2.render("",0,(255,255,255))
                                        Orden_ATC4=fuente2.render("",0,(255,255,255))
                                elif len(avionseleccionado.sprite.instrucciones)==3:
                                        Orden_ATC1=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC]),0,(255,255,255))
                                        Orden_ATC2=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC-
1]),0,(255,255,255))
                                        Orden_ATC3=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC-
2]),0,(255,255,255))
                                        Orden_ATC4=fuente2.render("",0,(255,255,255))
                                elif len(avionseleccionado.sprite.instrucciones)>=4:
                                        Orden_ATC1=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC]),0,(255,255,255))
                                        Orden_ATC2=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC-
1]),0,(255,255,255))
                                        Orden_ATC3=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC-
2]),0,(255,255,255))
                                        Orden_ATC4=fuente2.render(str(avionseleccionado.sprite.instrucciones[linea_conversacion_ATC-
3]),0,(255,255,255))

                        else:
                                destino_avion=fuente.render(" ",0,(255,255,255))
                                Velocidad_avion=fuente.render(" ",0,(255,255,255))
                                Estado_Avion=fuente.render(" ",0,(255,255,255))
                                Nombre_avion=fuente.render(" ",0,(255,255,255))
                                Orden_ATC1=fuente2.render(" ",0,(255,255,255))
                                Orden_ATC2=fuente2.render(" ",0,(255,255,255))
                                Orden_ATC3=fuente2.render(" ",0,(255,255,255))
                                Orden_ATC4=fuente2.render(" ",0,(255,255,255))
                        info_avio=fuente.render("- - - - - - - -  Información Avión  - - - - - - - -",0,(255,255,255))
                        Nivel_ASMGCS=fuente.render(TIPO_PROGRAMA,0,(100,100,100))
                        pantalla.blit(Nombre_avion,(1085,20))
                        pantalla.blit(info_avio,(966,300))
                        pantalla.blit(Nivel_ASMGCS,(20,20))
                        pantalla.blit(fuente.render("Cross",0,(255,255,255)),(botoncross.rect.left+95,botoncross.rect.top+8))
                        pantalla.blit(fuente.render("Hold short",0,(255,255,255)),(botonHS.rect.left+82,botonHS.rect.top+8))
                        pantalla.blit(fuente.render("Line up and wait",0,(255,255,255)),(botonLUAW.rect.left+63,botonLUAW.rect.top+8))
                        pantalla.blit(fuente.render("Cleared for takeoff",0,(255,255,255)),(botonCFT.rect.left+55,botonCFT.rect.top+8))
                        pantalla.blit(fuente.render("Número vuelo:",0,(255,255,255)),(980,330))
                        pantalla.blit(fuente.render("Destino:",0,(255,255,255)),(980,370))
                        pantalla.blit(fuente.render("Velocidad:",0,(255,255,255)),(980,410))
                        pantalla.blit(fuente.render("Estado:",0,(255,255,255)),(980,450))
                        pantalla.blit(fuente.render("Conversación ATC:",0,(255,255,255)),(965,490))
                        if len(avionseleccionado)>0:
                                pantalla.blit(fuente.render("knots",0,(255,255,255)),(1180,410))
```

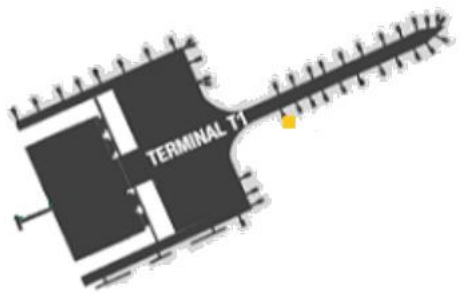
```
                    pantalla.blit(Nombre_avion,(1150,330))
                    pantalla.blit(destino_avion,(1150,370))
                    pantalla.blit(Velocidad_avion,(1150,410))
                    pantalla.blit(Estado_Avion,(1150,450))
                    pantalla.blit(Orden_ATC4,(967,527))
                    pantalla.blit(Orden_ATC3,(967,557))
                    pantalla.blit(Orden_ATC2,(967,587))
                    pantalla.blit(Orden_ATC1,(967,617))
                    for element in aviones:
                            fondo_nombre=pygame.Rect(element.rect.centerx+10,element.rect.centery-10,35,10)
                            pygame.draw.rect(pantalla,(94,91,91),fondo_nombre)

        pantalla.blit((pygame.font.Font(None,14)).render(element.nombre_avion,0,(255,255,255)),(element.rect.centerx+10,element.rect.centery-10))
                    pygame.display.flip()


if __name__ == '__main__':
    pygame.init()
    main()
```

ANEJO B: Manual funcionamiento programa

En este apartado se explica cómo genera una función el controlador a través de la interfaz del programa. Se detalla función, escenario que se debe dar y acción por parte del controlador.

Tabla 0.1 Funcionalidades programa

Botón	Función	Condiciones
Ponerse encima de un avión	Seleccionar avión	
	Nuevo destino 07L	Se debe tener un avión seleccionado
	Nuevo avión aterrizando por 07L	Avión no seleccionado Pista 07L/25R libre Algún finger disponible (destino)
	Nuevo destino 07R	Se debe tener un avión seleccionado
	Nuevo avión aterrizando por 07R	Avión no seleccionado Pista 07L/25R libre Algún finger disponible (destino)
	Nuevo destino 25L	Se debe tener un avión seleccionado
	Nuevo avión aterrizando por 25L	Avión no seleccionado Pista 07L/25R libre Algún finger disponible (destino)
	Nuevo destino 25R	Se debe tener un avión seleccionado
	Nuevo avión aterrizando por 25R	Avión no seleccionado Pista 07L/25R libre Algún finger disponible (destino)

	Dar paso al avión (“Avanzando”)	Avión seleccionado Ruta despejada
	Detener el avión (“Parado”)	Avión seleccionado
	Avance a pista (“Avanzando”)	Avión seleccionado Avión localizado punto acceso pista
	Despegar avión (“Despegando”)	Avión seleccionado Pista despejada
	Mirar instrucciones antiguas	Debe haber instrucciones previas
	Mirar instrucciones recientes	Debe haber instrucciones previas
	Nuevo avión desde finger	Avión no seleccionado Finger disponible
	Nuevo destino Finger seleccionado	Avión seleccionado Finger disponible
Botón derecho del ratón	Deseleccionar avión	Avión seleccionado
Botón “n”	Cambiar Nivel programa A-SMGCS	

Conclusiones

La seguridad en el transporte de la aviación es primordial y por tanto se debe aplicar una continua mejora de los sistemas de control y ayuda a todos los agentes involucrados. Todo soporte a uno de estos agentes (pilotos, controladores…) tendrá como resultado el mejor funcionamiento de los viajes en avión.

Centrándose en el objetivo de este proyecto, se ha concluido que el aumento continuo en el número de operaciones anuales que se realizan en el aeropuerto de Barcelona dificulta cada vez más la gestión de los movimientos en tierra de las diferentes aeronaves.

Para mejorar el sistema de control y distribución actual se deriva hacia la automatización de los procesos de cálculo de rutas y toma de decisiones. Para ello es necesario implantar progresivamente nuevas tecnologías que permitan incrementar las funcionalidades de los sistemas de las torres de control.

La implantación de estas nuevas herramientas puede ser un gran soporte para los controladores, especialmente en condiciones meteorológicas adversas donde se pierde capacidad de inspección visual.

Los beneficios de este proyecto se pueden desglosar en tres tipos: humanos, económicos y medioambientales. Los costes de estas nuevas tecnologías son elevados.

A pesar de los elevados costes, debido a que el proyecto es viable económicamente y a que los beneficios del proyecto se generan sobre muchos stakeholders diferentes (compañías aéreas, aseguradoras, estados…), se dan muchos posibles inversores iniciales y por tanto se podría conseguir la financiación con relativa facilidad.

Agradecimientos

Agradecimientos por su colaboración y ganas de difundir su conocimiento sobre la aviación al piloto Abel Sardaña.

Bibliografía

Referencias bibliográficas

[1] Plan Barcelona, *Nueva Terminal Sur*. 2003
http://www.aeropuertodebarcelona.net/index_archivos/documentos/historia/presentacion_terminal_sur.pdf

[2] Estadísticas tráfico aéreo español
http://www.aena.es/csee/Satellite?pagename=Estadisticas/Home

[3] Aeropuerto de Barcelona
https://es.wikipedia.org/wiki/Aeropuerto_de_Barcelona-El_Prat

[4] Información para víctimas y familiares de accidentes de Aviación Civil Comercial
http://www.aena.es/csee/ccurl/623/919/informacion%20victimas%20accidentes.pdf

[5] Informes provisionales accidentes e incidentes aéreos en España
https://www.fomento.gob.es/MFOM/LANG_CASTELLANO/ORGANOS_COLEGIADOS/CIAIAC/INVESTIGACION/2018/default.htm

[6] CIAIAC. Comisión de Investigación de Accidentes e Incidentes de Aviación Civil. Informe Anual 2016
http://www.fomento.es/NR/rdonlyres/B3609446-3DBD-4B54-91F8-6612315D0868/145349/Informeanual2016.pdf

[7] Información general aeropuerto Barcelona. Sistemas de control.
https://ext.eurocontrol.int/airport_corner_public/LEBL

[8] Especificaciones EUROCONTROL para los servicios avanzados del sistema de dirección y control del movimiento en la superficie (A-SMGCS)
http://www.eurocontrol.int/sites/default/files/content/documents/single-sky/specifications/edsp_17_003_spec_asmgcs_v1.0.pdf

[9] Normas y métodos recomendados internacionales. Anexo 14. Diseño y operaciones de aeródromos.

http://www.interairports.hn/wp-content/uploads/2015/08/Anexo-14-2009-Aerodromos.pdf